

INTERNATIONAL STANDARD

ISO
9506-1

Second edition
2003-08-01

Industrial automation systems — Manufacturing Message Specification —

Part 1: Service definition

*Systèmes d'automatisation industrielle — Spécification de messagerie
industrielle —*

Partie 1: Définition des services



Reference number
ISO 9506-1:2003(E)
Documents
Document Sharing Hub
© ISO 2003

PDF disclaimer

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

Adobe is a trademark of Adobe Systems Incorporated.

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

© ISO 2003

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

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

Published in Switzerland

Contents

Page

Foreword	xi
Introduction	xii
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	3
4 Abbreviations	6
5 Conventions	7
5.1 Base of Numeric Values	7
5.2 Object modelling	7
5.3 Specialisation of MMS	8
5.4 Service Parameter Description	9
5.5 Invocation Identifier on Service Primitives	11
5.6 List Of Modifier on Service Primitives	11
5.7 Addressing in MMS	11
5.8 Service Conventions	11
5.9 Calling and Called MMS-user	11
5.10 Sending and Receiving MMS-user and MMPM	12
5.11 Requesting and Responding MMS-user	12
5.12 Client and Server of a Service	12
5.13 Relationship of Object Models to Service Tables	13
6 MMS in the OSI Environment	13
6.1 Information Processing Tasks and Real Systems	13
6.2 Application Processes	13
6.3 Interaction of Application Processes	13
6.4 Interaction of Application Processes in OSI	14
6.5 Structure of Application Entities	14
6.6 Addressing of Application Entities	14
6.7 Application Context	15
6.8 Presentation Context, Abstract Syntaxes, and Transfer Syntaxes	15
6.9 MMS requirements of the communication system	15
7 The Virtual Manufacturing Device	21
7.1 Introduction	21
7.2 The Structure of a VMD	22
7.3 Transactions	30
7.4 Specification of Named objects	32
7.5 Object Name structure	35
7.6 Object Class structure	36
8 Environment And General Management services	36
8.1 Introduction and Models	36
8.2 Initiate service	45
8.3 Conclude service	51
8.4 Abort service	52
8.5 Cancel service	53
8.6 Reject service	55
9 Conditioned service response	59
9.1 Introduction and Models	59
9.2 AccessCondition parameter	65

9.3	DefineAccessControlList service	67
9.4	GetAccessControlListAttributes service	68
9.5	ReportAccessControlledObjects service	71
9.6	DeleteAccessControlList service	72
9.7	ChangeAccessControl service	73
10	VMD Support Services	77
10.1	Introduction	77
10.2	Status Response parameter	77
10.3	Status service	78
10.4	UnsolicitedStatus service	79
10.5	GetNameList service	79
10.6	Identify service	81
10.7	Rename service	82
10.8	GetCapabilityList service	84
10.9	VMDStop service	85
10.10	VMDReset service	86
11	Domain Management Services	87
11.1	Introduction and Models	87
11.2	InitiateDownloadSequence service	93
11.3	DownloadSegment service	95
11.4	TerminateDownloadSequence service	97
11.5	InitiateUploadSequence service	98
11.6	UploadSegment service	100
11.7	TerminateUploadSequence service	101
11.8	RequestDomainDownload service	102
11.9	RequestDomainUpload service	104
11.10	LoadDomainContent service	105
11.11	StoreDomainContent service	107
11.12	DeleteDomain service	110
11.13	GetDomainAttributes service	111
12	Program Invocation Management Services	113
12.1	Introduction and Models	113
12.2	CreateProgramInvocation service	120
12.3	DeleteProgramInvocation service	123
12.4	Start service	124
12.5	Stop service	128
12.6	Resume service	129
12.7	Reset service	132
12.8	Kill service	133
12.9	GetProgramInvocationAttributes service	134
12.10	Select service	137
12.11	AlterProgramInvocationAttributes service	139
12.12	ReconfigureProgramInvocation service	141
13	Unit Control	143
13.1	Introduction and Models	143
13.2	Control Element	144
13.3	InitiateUnitControlLoad service	145
13.4	UnitControlLoadSegment service	147
13.5	UnitControlUpload service	149
13.6	StartUnitControl service	151
13.7	StopUnitControl service	152
13.8	CreateUnitControl service	154
13.9	AddToUnitControl service	155
13.10	RemoveFromUnitControl service	156
13.11	GetUnitControlAttributes service	157
13.12	LoadUnitControlFromFile service	159
13.13	StoreUnitControlToFile service	160
13.14	DeleteUnitControl service	161
14	Variable Access Services	163

14.1	The MMS Variable Access Model	164
14.2	Specification of Types	171
14.3	Specification of Alternate Access	175
14.4	Specification of Data Values	179
14.5	Specification of Access to Variables	181
14.6	Read service	184
14.7	Write service	185
14.8	InformationReport service	187
14.9	GetVariableAccessAttributes service	188
14.10	DefineNamedVariable service	190
14.11	DeleteVariableAccess service	192
14.12	DefineNamedVariableList service	194
14.13	GetNamedVariableListAttributes service	196
14.14	DeleteNamedVariableList service	197
14.15	DefineNamedType service	199
14.16	GetNamedTypeAttributes service	201
14.17	DeleteNamedType service	202
14.18	Conformance	204
14.19	Guidance To Implementors	205
15	Data Exchange Management Services	206
15.1	The Data Exchange management model	206
15.2	ExchangeData service	207
15.3	GetDataExchangeAttributes service	209
16	Semaphore Management Services	210
16.1	The Semaphore Management Model	210
16.2	TakeControl service	217
16.3	RelinquishControl service	221
16.4	DefineSemaphore service	222
16.5	DeleteSemaphore service	224
16.6	ReportSemaphoreStatus service	226
16.7	ReportPoolSemaphoreStatus service	227
16.8	ReportSemaphoreEntryStatus service	229
16.9	AttachToSemaphore Modifier	232
16.10	Conformance	234
17	Operator Communication services	235
17.1	The Operator Communications Model	235
17.2	Input service	237
17.3	Output service	239
18	Event Management services	241
18.1	Event Detection and Notification	242
18.2	TriggerEvent service	247
18.3	EventNotification service	248
18.4	AcknowledgeEventNotification service	250
18.5	GetAlarmSummary service	252
18.6	GetAlarmEnrollmentSummary service	256
18.7	Attach To Event Condition Modifier	260
18.8	Conformance Requirements Unique to Event Management	262
19	Event Condition services	263
19.1	Event Conditions	263
19.2	DefineEventCondition service	268
19.3	DeleteEventCondition service	271
19.4	GetEventConditionAttributes service	273
19.5	ReportEventConditionStatus service	275
19.6	AlterEventConditionMonitoring service	277
20	Event Action services	279
20.1	Event Actions	279
20.2	DefineEventAction service	280
20.3	DeleteEventAction service	282

20.4	GetEventActionAttributes service	284
20.5	ReportEventActionStatus service	285
21	Event Enrollment services	287
21.1	Event Enrollments	287
21.2	DefineEventEnrollment service	295
21.3	DeleteEventEnrollment service	299
21.4	GetEventEnrollmentAttributes service	302
21.5	ReportEventEnrollmentStatus service	306
21.6	AlterEventEnrollment service	308
22	Event Condition List services	311
22.1	Event Condition Lists	311
22.2	DefineEventConditionList service	312
22.3	DeleteEventConditionList service	314
22.4	AddEventConditionListReference service	316
22.5	RemoveEventConditionListReference service	318
22.6	GetEventConditionListAttributes service	321
22.7	ReportEventConditionListStatus service	322
22.8	AlterEventConditionListMonitoring service	324
23	Journal Management services	326
23.1	The Journal Management Model	326
23.2	ReadJournal service	328
23.3	WriteJournal service	336
23.4	InitializeJournal service	339
23.5	ReportJournalStatus service	340
23.6	CreateJournal service	342
23.7	DeleteJournal service	343
23.8	Conformance Requirements Unique to Journals	344
24	Errors	344
24.1	Error Type	344
24.2	Description of structure of generic error type:	345
24.3	Additional Code	351
24.4	Additional Detail	351
24.5	Modifier Position	351
25	MMS Standardized Names	351
25.1	Introduction	351
25.2	Unique Name Assignment Mechanism	352
25.3	MMS Standardized Names	352
25.4	End of Module	355
26	Conformance	355
26.1	Introduction	355
26.2	Conformance Building Blocks (CBBs)	355
26.3	Static Conformance Requirements	357
26.4	Calling MMS-user Conformance Requirements	357
26.5	Called MMS-user Conformance Requirements	357
26.6	Server Conformance Requirements	357
26.7	Client Conformance Requirements	358
26.8	Parameter CBB Conformance Requirements	359
26.9	Dynamic Conformance	360
Annex A	(normative) Relationship of the VMD to an OSI Communication System	361
A.1	Introduction	361
A.2	Addressing of Application Entities	362
A.3	Conformance Requirements	362
Annex B	(normative) Requirements for Companion Standards	363
B.1	Introduction	363
B.2	Scope	363
B.3	Requirements	363

B.4	Outline of an MMS Companion Standard	363
Annex C (normative)	File Access service	369
C.1	ObtainFile service	369
Annex D (informative)	File Management services	371
D.1	Introduction	371
D.2	The MMS File Model	371
D.3	FileOpen service	373
D.4	FileRead service	374
D.5	FileClose service	375
D.6	FileRename service	376
D.7	FileDelete service	378
D.8	FileDirectory service	379
D.9	File Attributes parameter	381
D.10	Additional Specification for the Conclude Service	381
Annex E (informative)	Scattered Access	383
E.1	Introduction	383
E.2	Variable Specification parameter	385
E.3	DefineScatteredAccess service	386
E.4	GetScatteredAccessAttributes service	388
E.5	DeleteVariableAccess service	390
E.6	DefineNamedVariableList service	390
E.7	GetNamedVariableListAttributes service	391
E.8	DeleteNamedVariableList service	391
Annex F (informative)	MMS on TCP/IP	392
F.1	Introduction	392
F.2	General Internet Environments	392
F.3	References	392
	Index	393

Figures

Figure 1 - Relationships of Client and Server, Requesting and Responding MMS-user, and Sending and Receiving MPPM	12
Figure 2 - M-ASSOCIATE service	16
Figure 3 - M-RELEASE service	18
Figure 4 - M-DATA service	19
Figure 5 - M-U-ABORT service	20
Figure 6 - M-P-ABORT service	21
Figure 7 - Environment Management State Diagram	37
Figure 8 - Domain State Diagram	91
Figure 9 - Upload State Machines	93
Figure 10 - LoadDomainContent	105
Figure 11 - StoreDomainContent	108
Figure 12 - Program Invocation State Diagram	119
Figure 13 - Semaphore Entry model	215
Figure 14 - Token Semaphore model	216
Figure 15 - Pool Semaphore model	217
Figure 16 - Operator Station State Diagram	237
Figure 17 - Relationship Between Event Management Objects	242
Figure 18 - Network-triggered Event Condition State Diagram	266
Figure 19 - Monitored Event Condition State Diagram	267
Figure 20 - Event Action State Diagram	286
Figure 21 - State Diagram for &alarmAcknowledgmentRule = none	292
Figure 22 - State Diagram for &alarmAcknowledgmentRule = simple	293
Figure 23 - State Diagram for &alarmAcknowledgmentRule = ack-active	294
Figure 24 - State Diagram for &alarmAcknowledgmentRule = ack-all	295
Figure A.1 - The MMS Server Application Process	361
Figure D.1 - File Read State Machine	372

Tables

Table 1 - M-ASSOCIATE service	16
Table 2 - M-Release service parameters	18
Table 3 - M-Data service parameters	19
Table 4 - M-U-Abort service parameters	20
Table 5 - M-P-Abort service parameters	21
Table 6 - Local control	29
Table 7 - Name Class and Scope	33
Table 8 - Object Name	35
Table 9 - Object Class	36
Table 10 - Initiate service	45
Table 11 - Conclude service	51
Table 12 - Abort service	53
Table 13 - Cancel service	54
Table 14 - Reject service	55
Table 15 - Access Condition parameter	66
Table 16 - DefineAccessControlList service	67
Table 17 - GetAccessControlListAttributes service	69
Table 18 - ReportAccessControlledObjects service	71
Table 19 - DeleteAccessControlList service	72
Table 20 - ChangeAccessControl service	74
Table 21 - Status Response parameter	77
Table 22 - Status service	78
Table 23 - UnsolicitedStatus service	79
Table 24 - GetNameList service	80
Table 25 - Identify service	81
Table 26 - Rename service	83
Table 27 - GetCapabilityList service	84
Table 28 - VMDStop service	85
Table 29 - VMD attributes after VMDStop	86
Table 30 - VMDReset service	86
Table 31 - InitiateDownloadSequence service	94
Table 32 - DownloadSegment service	96
Table 33 - TerminateDownloadSequence service	97
Table 34 - InitiateUploadSequence service	99
Table 35 - UploadSegment service	100
Table 36 - TerminateUploadSequence service	101
Table 37 - RequestDomainDownload service	102
Table 38 - RequestDomainUpload service	104
Table 39 - LoadDomainContent service	106
Table 40 - StoreDomainContent service	108
Table 41 - DeleteDomain service	110
Table 42 - GetDomainAttributes service	111
Table 43 - CreateProgramInvocation service	120
Table 44 - DeleteProgramInvocation service	123
Table 45 - Start service	125
Table 46 - Stop service	128
Table 47 - Resume service	129
Table 48 - Reset service	132
Table 49 - Kill service	133
Table 50 - GetProgramInvocationAttributes service	135
Table 51 - Select service	138
Table 52 - AlterProgramInvocationAttributes service	140
Table 53 - ReconfigureProgramInvocation service	141
Table 54 - Control Element parameter	144
Table 55 - Interaction of Unit Control Primitives	146
Table 56 - InitiateUnitControlLoad service	146
Table 57 - UnitControlLoadSegment service	147
Table 58 - UnitControlUpload service	149
Table 59 - StartUnitControl service	151
Table 60 - StopUnitControl service	153

Table 61 - CreateUnitControl service	154
Table 62 - AddToUnitControl service	155
Table 63 - RemoveFromUnitControl service	156
Table 64 - GetUnitControlAttributes service	158
Table 65 - LoadUnitControlFromFile service	159
Table 66 - StoreUnitControlToFile service	160
Table 67 - DeleteUnitControl service	162
Table 68 - Type Description parameter	172
Table 69 - Type Specification parameter	175
Table 70 - Alternate Access parameter	176
Table 71 - Access Result parameter	179
Table 72 - Data parameter	180
Table 73 - Variable Access Specification parameter	181
Table 74 - Variable Specification parameter	182
Table 75 - Address parameter	183
Table 76 - Read service	184
Table 77 - Write service	186
Table 78 - InformationReport service	187
Table 79 - GetVariableAccessAttributes service	189
Table 80 - DefineNamedVariable service	190
Table 81 - DeleteVariableAccess service	192
Table 82 - DefineNamedVariableList service	194
Table 83 - GetNamedVariableListAttributes service	196
Table 84 - DeleteNamedVariableList service	198
Table 85 - DefineNamedType service	200
Table 86 - GetNamedTypeAttributes service	201
Table 87 - DeleteNamedType service	203
Table 88 - DataExchange service	207
Table 89 - GetDataExchangeAttributes service	209
Table 90 - TakeControl service	217
Table 91 - RelinquishControl service	221
Table 92 - DefineSemaphore service	223
Table 93 - DeleteSemaphore service	225
Table 94 - ReportSemaphoreStatus service	226
Table 95 - ReportPoolSemaphoreStatus service	228
Table 96 - ReportSemaphoreEntryStatus service	230
Table 97 - AttachToSemaphore Modifier	232
Table 98 - Input service	238
Table 99 - Output service	240
Table 100 - TriggerEvent service	247
Table 101 - EventNotification service	248
Table 102 - AcknowledgeEventNotification service	251
Table 103 - GetAlarmSummary service	253
Table 104 - GetAlarmEnrollmentSummary service	257
Table 105 - Attach To Event Condition Modifier	261
Table 106 - DefineEventCondition service	268
Table 107 - DeleteEventCondition service	271
Table 108 - GetEventConditionAttributes service	273
Table 109 - ReportEventConditionStatus service	276
Table 110 - AlterEventConditionMonitoring service	277
Table 111 - DefineEventAction service	281
Table 112 - DeleteEventAction service	282
Table 113 - GetEventActionAttributes service	284
Table 114 - ReportEventActionStatus service	285
Table 115 - DefineEventEnrollment service	296
Table 116 - DeleteEventEnrollment service	299
Table 117 - GetEventEnrollmentAttributes service	302
Table 118 - ReportEventEnrollmentStatus service	307
Table 119 - AlterEventEnrollment service	309
Table 120 - DefineEventConditionList service	313
Table 121 - DeleteEventConditionList service	315
Table 122 - AddEventConditionListReference service	316
Table 123 - RemoveEventConditionListReference service	319
Table 124 - GetEventConditionListAttributes service	321

Table 125 - ReportEventConditionListStatus service	322
Table 126 - AlterEventConditionListMonitoring service	324
Table 127 - ReadJournal service	329
Table 128 - WriteJournal service	336
Table 129 - InitializeJournal service	339
Table 130 - ReportJournalStatus service	341
Table 131 - CreateJournal service	342
Table 132 - DeleteJournal service	343
Table 133 - Structure of Error Type	345
Table C.1 - ObtainFile service	369
Table D.1 - FileOpen service	373
Table D.2 - FileRead service	374
Table D.3 - FileClose service	376
Table D.4 - FileRename service	377
Table D.5 - FileDelete service	378
Table D.6 - FileDirectory service	379
Table D.7 - File Attributes parameter	381
Table E.1 - Scattered Access Description parameter	386
Table E.2 - DefineScatteredAccess service	387
Table E.3 - GetScatteredAccessAttributes service	389

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 matter of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

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

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

ISO 9506-1 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC 5, *Architecture, communications and integration frameworks*.

This second edition of ISO 9506-1 cancels and replaces the first edition (ISO 9506-1:2000), of which it constitutes a technical revision. It incorporates several technical corrections to ISO 9506-1:2000. The first edition of ISO 9506-1:2000 included corrections published in ISO/IEC 9506-1/Cor.1:1995 and in ISO/IEC 9506-1/Cor.2:1995, the additional services published in ISO/IEC 9506-1/Amd.1:1993, and in ISO/IEC 9506-1/Amd.2:1995, and the material published in ISO/TR 13345.

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

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

Introduction

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

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

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

Security considerations

When implementing MMS in secure or safety critical applications, features of the OSI security architecture may need to be implemented. This International Standard provides simple facilities for authentication (passwords) and access control. Systems requiring a higher degree of security will have to consider features beyond the scope of this International Standard. This International Standard does not use facilities for non-repudiation.

Complexity of services and requirements

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

Keywords

Application Interworking
Application Layer Protocol
Information Processing Systems
Manufacturing Communications Network
Manufacturing Message Specification
Numerical Control System
Open Systems Interconnection

OSI Reference Model
Process Control System
Programmable Controller
Programmable Device
Robotics Control System
Virtual Manufacturing Device

General

This part of ISO 9506 is one of a set of 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 9506 is to define the services provided by the Manufacturing Message Specification. The Services specified in this part of ISO 9506 are realized by the Manufacturing Message

Specification Protocol specified in ISO 9506-2, making use of services available from the underlying communication system. Although using the model in ISO 7498, this International Standard is written to be independent of the exact form of the communication system insofar as possible. This International Standard does this by describing its support requirements as a set of abstract services present in a variety of communication environments. Realization of these abstract services in an OSI environment is described in Annex A.

This part of ISO 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 9506-2 specifies the protocol that supports the Manufacturing Message Specification.

Edition

This part of ISO 9506 differs from the first edition of ISO 9506-1 by correcting several errors in the ASN.1 type definitions and modelling structures. It also corrects several typographical errors in that document.

This part of ISO 9506 differs from ISO/IEC 9506-1:1990 in the following ways.

- a) The informal object modelling used in ISO/IEC 9506-1 has been replaced by the use of the modelling techniques present in ASN.1, ISO/IEC 8824-2. Hence, this part of ISO 9506 defines an ASN.1 module, MMS-Object-Module-1, that contains the object models on which the service procedures are based.
- b) The material in ISO/IEC TR 13345 that specifies subsets of protocol for MMS has been used in this part of ISO 9506 to specify options within the object models. The material in ISO/IEC TR 13345 has been modified slightly to allow for use of the PER encoding rules, ISO/IEC 8825-2.
- c) All the material of Amendments 1 and 2 have been incorporated into the document, as well as the Technical Corrigenda.
- d) The services and protocol present in the Companion Standards already published, ISO/IEC 9506-3, ISO/IEC 9506-4, and ISO/IEC 9506-6, have been incorporated into the base standard, and new parameter CBBs have been added to the Initiate procedure to indicate their presence. The concept of Companion Standard has been simplified to a document that makes explicit the relationship between the abstract models in MMS and the requirements of the application field that is the subject of the Companion Standard.

As a result of this incorporation, the need for separate abstract syntaxes for each of the Companion Standards has been removed. All Companion Standards can now operate in the single abstract syntax of the base standard, although using other abstract syntaxes remains a possibility for backward compatibility.

- e) The communication requirements of MMS have been generalized so that MMS is described with respect to an abstract set of services needed for its support. The relation between this abstract set of services and the services provided by the suite of OSI communication protocols is specified in an annex. This opens the possibility of having MMS operating correctly over alternate communication systems (such as reduced stack implementations) as long as the equivalent of these abstract services is provided.
- f) The restrictions on the characters that can be used as an Identifier have been relaxed to allow an Identifier to begin with a numeric character, and by extension, to consist solely of numeric characters.
- g) Many (but not all) occurrences of VisibleString have been replaced by a new production MMSString that provides the option of using an arbitrary string of characters taken from ISO 10646. Similarly, these more general strings may also be used as Identifiers. A new parameter CBB has been added to provide for negotiation of the use of these more general strings.
- h) A new service, ReconfigureProgramInvocation, has been introduced into the clause on Program Invocation management. This service provides a technique of dynamically changing the constituent Domains of a running Program Invocation.
- i) A new field has been added to the object model of the Named Variable and Named Type. This field may be used to describe the semantics associated with the Named Variable or Named Type. The field is either predefined or has its value established as the name of the Named Type used to construct it in the DefineNamedVariable or DefineNamedType service. This field can be reported with the

GetVariableAccessAttributes or GetNamedTypeAttributes service if a parameter CBB appropriate to this field has been negotiated.

- j) The material of the document has been reorganized to provide shorter clauses.
- k) The Real Data type has been removed from the document.
- l) The Scattered Access has been removed from the base document and placed in an informative annex.
- m) In accordance with the recommendations in ISO/IEC 8824-1, all occurrences of EXTERNAL in the protocol have been replaced with CHOICE { EXTERNAL, EMBEDDED PDV }.

Protocol

Because of the use of the ASN.1 object modelling technique, the protocol now exists in two separate modules, one that is part of the object model contained in this part of ISO 9506, and a second module defined in ISO 9506-2 that describes the content and structure of all valid PDUs. Despite the fact that the ASN.1 formulation appears different in some cases, nevertheless the PDUs produced through application of the first edition of ISO/IEC 9506 are identical with those produced by this edition. For this reason, this edition continues to be identified by the major version number one. (The minor version number has been changed to reflect all the new additions to the document.)

There are exceptions to this statement that should be noted.

- a) Syntactic extensions defined by the companion standards are now identified by new parameter CBBs instead of a separate abstract syntax. Therefore, for any use of MMS involving companion standard facilities, there is a change in the Initiate PDU. However, if the companion standard facilities are not used, the Initiate PDU remains the same as that defined by the first edition.
- b) Some small changes have been made to the tagging in the ChangeAccessControl service (part of Amendment 2) to bring it into alignment with corresponding protocol in the GetNameList and Rename services.
- c) Encoding of the PDUs using PER (ISO/IEC 8825-2) may not be completely compatible with PDUs generated by the first edition of ISO/IEC 9506:1990; this is because replacement of a type by a CHOICE containing that type will result in a different encoding using PER; BER encoding for these two situations is identical. Thus, if the PDUs contain any elements that are EXTERNAL, according to item m) above, they will be replaced by a CHOICE resulting in a different PER encoding.

ASN.1 Modules

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

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

Industrial automation systems — Manufacturing Message Specification —

Part 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 9506 as the manufacturing environment. This part of ISO 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 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 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 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 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 9506 specifies the externally visible functionality of implementations together with conformance requirements for such functionality.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 646:1991,	<i>Information technology - ISO 7-bit coded character set for information interchange</i>
ISO/IEC 7498-1:1994,	<i>Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model</i>
ISO 7498-2:1989,	<i>Information processing systems - Open Systems Interconnection - Basic Reference Model: Security Architecture</i>
ISO 7498-3:1997,	<i>Information technology - Open Systems Interconnection - Basic Reference Model: Naming and addressing</i>

ISO 9506-1: 2003(E)

ISO 8571 (all parts),	<i>Information processing systems - Open Systems Interconnection - File Transfer, Access and Management</i>
ISO/IEC 8649:1996,	<i>Information technology - Open Systems Interconnection - Service definition for the Association Control Service Element</i>
ISO/IEC 8650-1:1996,	<i>Information technology - Open Systems Interconnection - Connection-oriented protocol for the Association Control Service Element: Protocol specification</i>
ISO 8822:1994,	<i>Information technology - Open Systems Interconnection - Presentation service definition</i>
ISO/IEC 8824-1:1998,	<i>Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation</i>
ISO/IEC 8824-2:1998,	<i>Information technology - Abstract Syntax Notation One (ASN.1): Information object specification</i>
ISO/IEC 8825-1:1998,	<i>Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)</i>
ISO/IEC 8825-2:1998,	<i>Information technology - ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)</i>
ISO 9506-2 ¹⁾ ,	<i>Industrial automation systems - Manufacturing Message Specification - Part 2: Protocol specification</i>
ISO/IEC 9545:1994,	<i>Information technology - Open Systems Interconnection - Application Layer structure</i>
ISO/IEC 10646-1:2000,	<i>Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane</i>
ISO/IEC 10731:1994,	<i>Information technology - Open Systems Interconnection - Basic Reference Model - Conventions for the definition of OSI Services</i>
ANSI/IEEE 754:1985,	<i>IEEE Standard for Binary Floating-Point Arithmetic</i>

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

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

3.1 Reference Model definitions

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

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

¹⁾ To be published.

3.2 Service Convention definitions

This part of ISO 9506 makes use of the following terms defined in the Conventions for the definition of OSI Services (ISO/IEC 10731) as they apply to the Manufacturing Message Specification:

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

3.3 Abstract Syntax Notation definitions

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

- | | |
|------------------------------------|----------------------------|
| a) value; | o) octetstring type; |
| b) type; | p) null type; |
| c) simple type; | q) sequence type; |
| d) structure type; | r) sequence-of type; |
| e) component type; | s) tagged type; |
| f) tag; | t) choice type; |
| g) tagging; | u) selection type; |
| h) type (or value) reference name; | v) object identifier type; |
| i) character string type; | w) module; |
| j) boolean type; | x) production; |
| k) true; | y) ASN.1 encoding rules; |
| l) false; | z) ASN.1 character set; |
| m) integer type; | aa) external type. |
| n) bitstring type; | |

3.4 Other definitions

For the purposes of this document, the following terms and definitions also apply.

3.4.1

AA-specific (Application Association specific):

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

Called MMS-user:

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

3.4.4

Calling MMS-user:

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

3.4.5

Client:

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

3.4.6

conformance building block (CBB):

an atomic unit used to describe MMS conformance requirements.

3.4.7

data:

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

3.4.8

domain:

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

3.4.9

Domain-specific:

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

3.4.10

download:

the process of transferring the content of a Domain, including any subordinate objects, via load data to a remote user.

3.4.11

event management:

the management of event conditions, event actions, event enrollments, and event condition lists.

3.4.12

file:

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

3.4.13

file operation:

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

3.4.14

filestore:

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

3.4.15

information:

the combination of data and the meaning that it conveys.

3.4.16

journal:

a set of recorded, time-tagged event transitions, variable data, and/or comments, that 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 9506.

3.4.18**Manufacturing Message Protocol Machine (MMPM):**

an abstract machine that carries out the procedures specified in this part of ISO 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 part of the application process that 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 that occurs due to an explicit stimulus 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 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 that most closely corresponds to an execution thread in a multi-tasking environment, 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**Requesting MMS-user:**

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

3.4.30**Responding MMS-user:**

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

3.4.31**semaphore:**

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

3.4.32**semaphore management:**

the control of semaphores.

3.4.33

Sending MMPM:

the MMPM that sends an MMS PDU.

3.4.34

Sending MMS-user:

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

3.4.35

Server:

the peer communicating entity that behaves as an agent for a VMD for a particular service request instance.

3.4.36

standardized object:

an object instantiation whose definition is provided in this part of ISO 9506 or in an MMS Companion Standard.

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

Virtual Manufacturing Device (VMD):

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

3.4.41

VMD-specific:

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

4 Abbreviations

AA	application association
ACSE	Association Control Service Element
AE	application entity
AP	application process
ASE	application service element
ASN.1	Abstract Syntax Notation One
CBB	conformance building block
CIS	Configuration and Initialization Statement
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 9506 uses a decimal representation for all numeric values unless otherwise noted.

5.2 Object modelling

This part of ISO 9506 makes use of a technique of abstract object modelling in order to describe fully 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 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 9506.

MMS defines several classes of objects. Each object is an instance of a class; a class 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 that 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 9506 or may be established by MMS services; hence a change in the device may be modelled 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 that is a part of this combination of attributes that make the object unique is identified as a "key attribute".

Some attributes of the objects only need occur if certain parameters are negotiated during association establishment. These parameters, exchanged during the association establishment procedure, are called conformance building blocks (CBB). While the basic object model remains unaffected, the exact details of the object model in use on any instance of communication depends on which of these parameters have been negotiated during association establishment.

Finally, some objects contain attributes that are conditional, in the sense that they are relevant to the object if and only if certain conditions (other than the conformance building blocks) hold true. MMS expresses such attributes through the use of a "constraint". 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.

This International Standard makes use of the object modelling facilities of ASN.1, ISO/IEC 8824-2. The type language specified in ISO/IEC 8824-1 is used for describing the abstract structure of a protocol, that is, the data present in a message. It does so by providing a notation for data values and data types. A data value is an instance of a data type, and a data type can be thought of as a name for the set of possible data values for that type.

ISO/IEC 8824-2 extends this paradigm to introduce the concept of object and class. An object is an instance of class, and a class is defined by an abstract structure of a set of attributes, called fields. Each field is composed of two elements, a field identifier, and a field type. For the purposes of MMS models, the field can be restricted to one of four possible choices:

- a) the field is a data value (an instance of a data type);
- b) the field is a data value set (a subtype of a data type);
- c) the field is an object (an instance of a class);
- d) the field is an object set (a subset of a class).

The field identifier always begins with an ampersand (&) followed by a name. The use of capital and lower case letters serve to specify which choice is being used. The following example illustrates the four possibilities:

```
NewObject ::= CLASS {
    &invokeID      INTEGER,
    &Capabilities  MMSString,
    &selected-PI   PROGRAM-INVOCATION,
    &AbstractSyntaxes ABSTRACT-SYNTAX }
```

- &invokeID. The field identifier begins with a lower case letter (the 'i' following the ampersand) and the field choice is the name of a data type, INTEGER. This field is a simple data value field.
- &Capabilities. The field identifier begins with an upper case letter and the field choice is the name of a data type. This field is a value set, one or more values from a specific type.
- &selected-PI. The field choice identifies an object class, a PROGRAM-INVOCATION. By convention, all object classes are named using capital letters. The field identifier itself begins with a lower case letter indicating that the field is a single object from that class.
- &AbstractSyntaxes. The field identifier begins with an upper case letter and the field choice identifies an object class. This field is an object set composed of one or more instances of the object class ABSTRACT-SYNTAX. (ABSTRACT-SYNTAX is an object class defined in an annex to ISO/IEC 8824-2.)

NOTE ISO/IEC 8824-2 allows other specifications for fields, but these are the only ones that are used in MMS models.

5.2.1 References to other objects

Some objects contain attributes that identify other objects. In the example above, &selected-PI and &AbstractSyntaxes identify other objects. Such attributes, called reference attributes, provide a mechanism to create a linkage from one object to another. The method of representing 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.

In some cases, references to other objects is indicated by a field that contains a value of a keyword of the referenced object (in most cases, an Identifier or ObjectName). The object so referenced will be indicated in a comment.

In certain cases, reference attributes may be missing from the object model. This possibility, an empty set of values or objects, is indicated in the object model by the keyword OPTIONAL following the field choice.

5.2.2 Field Types

Types for fields defined in this part ISO 9506 make use of types defined in ISO/IEC 8824-1, clause 3. Additionally, complex types are used that are constructed from the ISO/IEC 8824-1 primitive types and named to allow them to be referenced.

5.3 Specialisation of MMS

5.3.1 Conformance Building Blocks (CBB)

MMS prescribes a procedure, used at the time of association establishment, in which sets of parameters are exchanged for the purpose of identifying the services that may be performed during the association. The effect of identifying these parameters is such that the protocol available to be used during the association is limited to a proper subset of the entire protocol specified in ISO 9506-2. This same procedure affects the attribute structure of the object model operative for that instance of communication. The parameters exchanged are of two types:

- those that are announced by the two MMS users, the service conformance building blocks,
- those that are negotiated between the two MMS-users, the parameter conformance building blocks. These CBBs are proposed by the Calling MMS-user and either accepted or rejected by the Called MMS-user. Negotiation always works to reduce the set proposed, never to augment it.

Declaration of support of any service CBB requires inclusion of the protocol related to that service in the protocol set to be used on that association. Support of a parameter CBB usually results in the inclusion of some optional fields within the protocol of some service request or response. However, in some cases, support of a parameter

CBB implies support of one or more service CBBs, regardless of whether or not support for those service CBBs has been declared. The full effect of these CBBs on the protocol specification is given in ISO 9506-2. Here we note the effect of these CBBs on the object model.

5.3.2 Notation

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

- IF (<list of arguments>)
- ELSE
- ENDIF.

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

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

IF (x)

IF (y)

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

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

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

5.3.3 Constraints

Constraints (other than occurrences of CBBs) are expressed in this object model by use of the ASN.1 construct OPTIONAL together with a comment that indicates the conditions under which this attribute is or is not included in the object model.

5.4 Service Parameter Description

This part of ISO 9506 uses a tabular format to describe the component parameters of the MMS service primitives. This same format is used to describe several complex parameters that are used in more than one service. 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 if the service is not a confirmed service. For parameter specifications, the Req and Rsp may be combined into a single column as may the Ind and Cnf.

5.4.1 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
- 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, 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, there is no dependency on other conformance building blocks. If an entry does exist, the parameter is available (and permitted for use by this International 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 9506-2.)

The descriptions of parameters in this part of ISO 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/IEC 8824-1 or may be defined in ISO 9506-2.

5.4.2 Collating Sequences

Several object models contain fields that are themselves sets of other objects (more properly sets of references to other objects). When such sets are to be reported as parameters of a service response, the order of the elements shall be based on the &name field of the objects. The syntax of this &name field is defined by the Identifier type in clause 7.5.2 of ISO 9506-2. The Identifier type lists the permissible characters of the &name field, and gives an ordering of these characters. This ordering shall be used to define the collating sequence of the set of objects so named.

If objects of different name scopes are to be combined on a composite list, the objects of AA-specific scope shall appear first grouped by the application association, the objects of VMD specific scope shall appear next, and the objects of Domain scope shall appear last grouped by Domain. The ordering of the Domains is as specified in this clause; the ordering of the application associations is a local matter.

Some service parameters refer to arbitrary sequences of character strings (MMSString); in the cases in which a collating sequence is needed for these parameters, the character strings should be ordered using the prescriptions of Clause 36 of ISO/IEC 8824-1.

5.5 Invocation Identifier on Service Primitives

For services identified in the ConfirmedServiceRequest production in clause 7.1 of ISO 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.

5.6 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 that 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 16 and 18, respectively.

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

5.7 Addressing in MMS

This International Standard does not provide the means for naming and addressing of a peer MMS-user or peer Manufacturing Message Protocol Machine (MMPM). It makes use of the addressing and naming facilities in the underlying communication system. Specifics about the use of naming and addressing within OSI may be found in Annex A.

5.8 Service Conventions

This part of ISO 9506 uses the descriptive conventions contained in the Conventions for the definition of OSI Services (ISO/IEC 10731). The Conventions for the definition of OSI Services 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/IEC 10731 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.

5.9 Calling and Called MMS-user

This part of ISO 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.

5.10 Sending and Receiving MMS-user and MMPM

This part of ISO 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 some time. The first MMS-user sends the request and receives the confirmation, while the second MMS-user receives the indication and sends the response.

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

5.11 Requesting and Responding MMS-user

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

5.12 Client and Server of a Service

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

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.

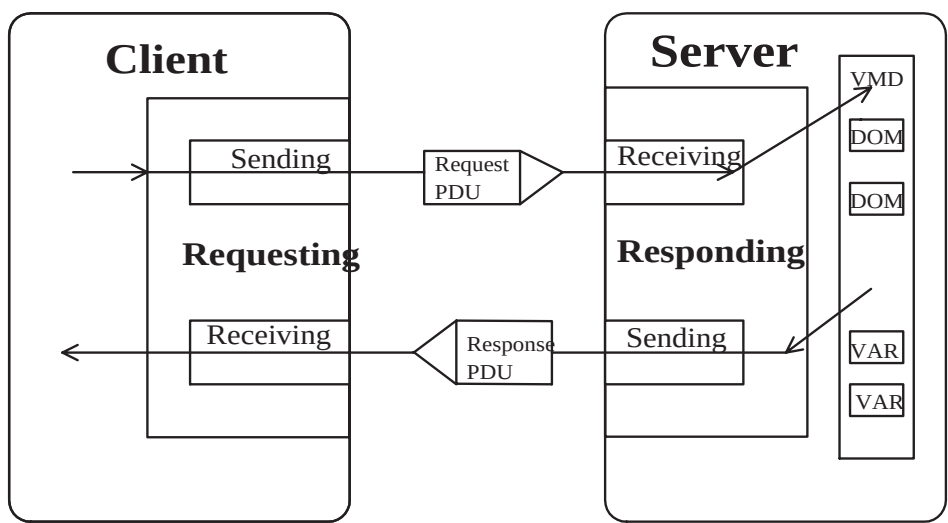


Figure 1 - Relationships of Client and Server, Requesting and Responding MMS-user, and Sending and Receiving MMPM

5.13 Relationship of Object Models to Service Tables

Throughout the text, the term "parameter" will be used to refer to components of a service primitive, and the term "attribute" or "field value" will refer to a component of an object model. In many cases, there is a one to one correspondence between a parameter in a service primitive and an attribute in an object model. However, in other cases the parameter is derived from one or more object attributes. When referring to a distinguished value of an attribute in an object model, a **distinctive type font** will be used. When referring to a distinguished value of a parameter of a service primitive, the value will be in upper case, e.g. UNDEFINED.

6 MMS in the OSI Environment

This clause describes the relationship between MMS and the communication environment that supports it. Although this International Standard is designed to be used in a variety of communication environments, it uses the model and terms of the OSI model (ISO 7498 and ISO/IEC 10731) to describe its environment and the elements of the services needed to support it. Specific use of OSI communications is defined in annex A. Clause 7 describes the specific model of the MMS device within this environment.

Additional information to supplement this clause may be found in the OSI Reference Model (ISO 7498-1 and ISO 7498-3), the Conventions for the definition of OSI Services (ISO/TR 8509), and the OSI Application Layer Structure (ISO/IEC 9545). References may be found in clause 2. This information may be particularly useful in understanding the relationship between AEs, APs, and their respective invocations.

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 7 to 23, 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 that behaves externally as a VMD can be considered to be in conformance with ISO 9506.

NOTE 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.

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 this task successfully, 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 follow the OSI Reference model.

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 that 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 that 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 these aspects of the

"world" that 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 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 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 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.

6.6 Addressing of Application Entities

The underlying communication system uses some system of addressing and naming to identify nodes on the communication network and AE-invocations within each node. In the process of establishing an application association to support the MMS environment, the MMS-user must have knowledge of the addressing conventions

used by the underlying communication. However, the specification of this addressing and naming conventions is outside the scope of this International Standard.

While in the MMS environment, an MMS user may need to refer to some communication node and/or AE-invocation other than the one with which the application association has been established. For this purpose this International Standard makes use of an "Application Reference" to identify an AE in another system. This is in the form of an ASN.1 type definition "ApplicationReference", appropriate to the underlying communication system. For use of MMS in an OSI based communication system, ApplicationReference is defined in annex A.

6.7 Application Context

The application context 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 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 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 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 that 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. A specific combination of an abstract syntax and a transfer syntax that is used for transfer of data 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/IEC 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 Basic Encoding Rules (ISO/IEC 8825-1) to the MMS abstract syntax, but others are possible. The only requirement on a transfer syntax to be used for MMS is that it faithfully encode all of the elements of abstract syntax defined ISO 9506-2.

6.9 MMS requirements of the communication system

This International Standard requires that the underlying communication system provide the encoding mechanism (transfer syntax) to support the transfer of data expressed in the abstract syntax defined in ISO 9506-2. This International Standard requires the communication system to support full duplex, connection oriented peer to peer communication. In addition, this International Standard requires the support of the following abstract services from the communication system. The relationship of these services to OSI communication facilities is described in annex A, and the use of these services by MMS services is described in clause 24 of ISO 9506-2.

6.9.1 M-ASSOCIATE

The M-ASSOCIATE service allows an MMS-user to establish an association with another MMS-user over the communication system and to convey to the peer MMS-user a set of MMS specific parameters to be associated with this instance of communication. This service requires a set of parameters most of which are specific to the communication systems being employed. However, there are two parameters required of this service in support of other MMS functions, the identity of the other MMS-user, and (optionally) a password or other authentication parameter to be transmitted to the peer MMS-user. M-ASSOCIATE is a confirmed service. In addition to its other functionality, M-ASSOCIATE allows one MMS-user to transmit some user data (the service parameters of the

MMS Initiate service) to its peer MMS-user. Upon completion of the service either (1) an association has been established and the MMS environment has been created (positive result parameter), or (2) the association could not be established (negative result parameter). Figure 2 shows the time sequence of service primitives for this service.

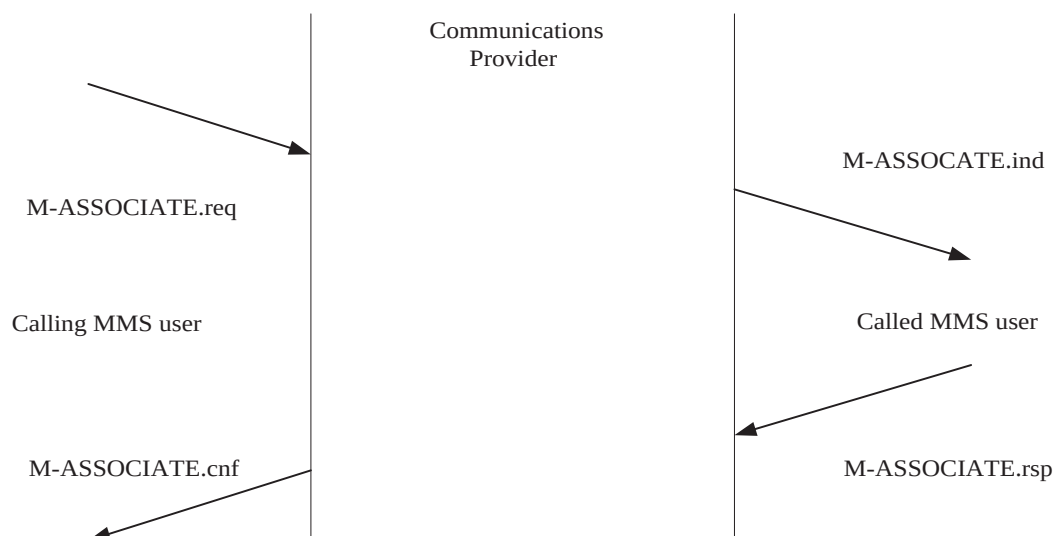


Figure 2 - M-ASSOCIATE service

Table 1 shows the parameters of the M-ASSOCIATE service.

Table 1 - M-ASSOCIATE service

Parameter	Req	Ind	Rsp	Cnf
Argument	M	M		
Calling Application Reference	M	M(=)		
Called Application Reference	M	M(=)		
Other Communication Parameters	M	M		
Authentication Value	U	U(=)		
User Data	M	M(=)		
Result			M	M(=)
Responding Application Reference			M	M(=)
Other Communication Parameters			M	M
Authentication Value			U	U(=)
User Data			M	M(=)

6.9.1.1 Argument

This parameter contains the parameters of the M-ASSOCIATE service of the Calling system.

6.9.1.1.1 Calling Application Reference

This parameter identifies the calling system uniquely within the communication system.

6.9.1.1.2 Called Application Reference

This parameter identifies the system with which the Calling system wishes to establish an association.

6.9.1.1.3 Other Communication Parameters

This parameter is one or more parameters specific to the underlying communication system being used.

6.9.1.1.4 Authentication Value

This optional parameter may be included if the called system requires authentication to allow some services to be performed. It is used to establish the right of the calling system to make use of those services.

6.9.1.1.5 User Data

This parameter contains the MMSPdu as formed by the Initiate service request (see clause 8).

6.9.1.2 Result

This parameter shall indicate whether the M-ASSOCIATE service request was accepted or rejected.

6.9.1.2.1 Responding Application Reference

This parameter identifies the system that is responding to the M-ASSOCIATE service request. It will normally be identical to the Called Application Reference of the service request, but it need not be so.

6.9.1.2.2 Other Communication Parameters

This parameter is one or more parameters specific to the underlying communication system being used.

6.9.1.2.3 Authentication Value

This optional parameter may be included if the calling system requires authentication to allow some services to be performed. It is used to establish the right of the called system to make use of those services.

6.9.1.2.4 User Data

This parameter contains the MMSPdu as formed by the Initiate service response, (see clause 8).

6.9.2 M-RELEASE

The M-RELEASE service allows an MMS-user to terminate an association in an orderly manner. M-RELEASE is a confirmed service. In addition to its other functionality, M-RELEASE allows one MMS-user to transmit some user data (the service parameters of the MMS Conclude service) to its peer MMS-user. Upon completion of the service either (1) the association has been terminated and the MMS environment has been exited (positive result parameter), or (2) the association could not be terminated (negative result parameter). Figure 3 shows the time sequence of service primitives for this service.

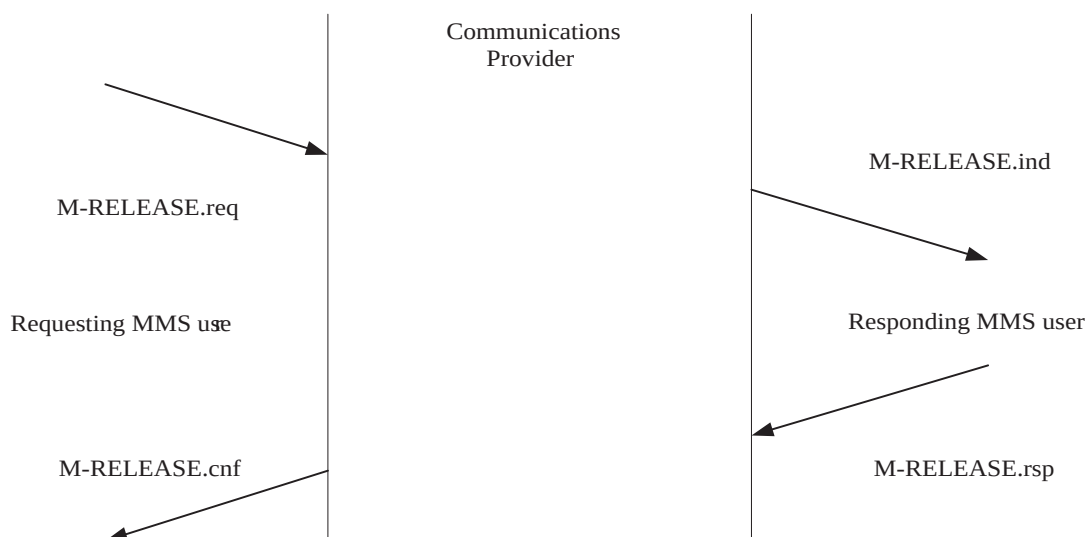


Figure 3 - M-RELEASE service

Table 2 shows the parameters of the M-RELEASE service.

Table 2 - M-Release service parameters

Parameters	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Other Communication Parameters	M	M		
User Data	M	M(=)		
Result			M	M(=)
Other Communication Parameters			M	M
User Data			M	M(=)

6.9.2.1 Argument

This parameter contains the parameters of the M-RELEASE service of the requesting system.

6.9.2.1.1 Other Communication Parameters

This parameter is one or more parameters specific to the underlying communication system being used.

6.9.2.1.2 User Data

This parameter contains the MMSPdu as formed by the Conclude service request (see clause 8).

6.9.2.2 Result

This parameter shall indicate whether the M-RELEASE service request was accepted or rejected.

6.9.2.2.1 Other Communication Parameters

This parameter is one or more parameters specific to the underlying communication system being used.

6.9.2.2.2 User Data

This parameter contains the MMSPdu as formed by the Conclude service response, (see clause 8).

6.9.3 M-DATA

The M-DATA allows an MMS-user to transfer an arbitrary amount of data to its peer MMS-user. Most MMS services make use of this service. M-DATA is an unconfirmed service. The principal functionality of M-DATA is to allow one MMS-user to transmit some user data (the service parameters of the MMS confirmed and unconfirmed services) to its peer MMS-user. Figure 4 shows the time sequence of service primitives for this service.

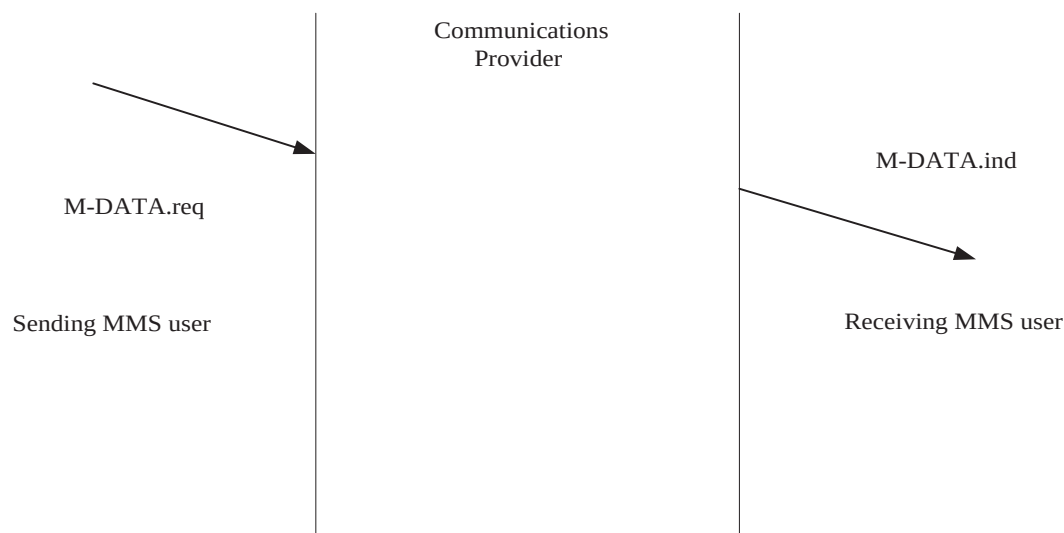


Figure 4 - M-DATA service

Table 3 shows the parameters of the M-DATA service.

Table 3 - M-Data service parameters

Parameters	Req	Ind
Argument User Data	M M	M M(=)

6.9.3.1 Argument

This parameter contains the parameters of the M-DATA service of the requesting system.

6.9.3.1.1 User Data

This parameter contains the MMSPdu as formed by the MMS service request or response.

6.9.4 M-U-ABORT

The M-U-ABORT service allows one MMS-user to abruptly terminate the association. The M-U-ABORT service carries no user data. Figure 5 shows the time sequence of service primitives for this service.

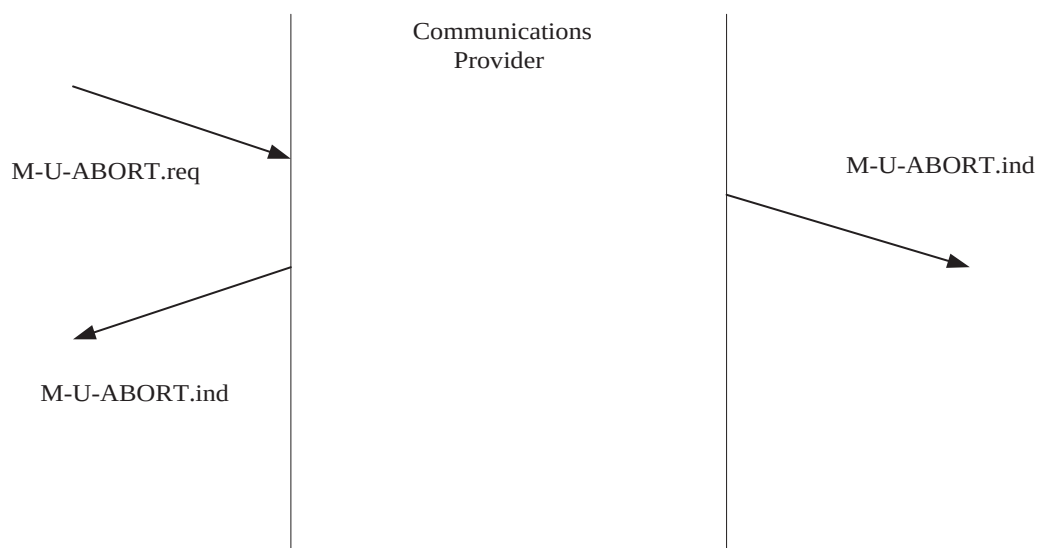


Figure 5 - M-U-ABORT service

Table 4 shows the parameters of the M-U-Abort service.

Table 4 - M-U-Abort service parameters

Parameters	Req	Ind
Argument Abort Source	M	M M

6.9.4.1 Argument

This parameter contains the parameters of the M-U-ABORT service of the requesting system.

6.9.4.1.1 Abort Source

This parameter is present in the indication and indicates the source of the abort request. If true, the abort was requested from the local system; if false, the abort was requested from the remote system.

6.9.5 M-P-ABORT

The M-P-ABORT service provides indication to the MMS-user that the communication system has abruptly terminated the association. The M-P-ABORT service carries no user data. Figure 6 shows the time sequence of service primitives for this service.

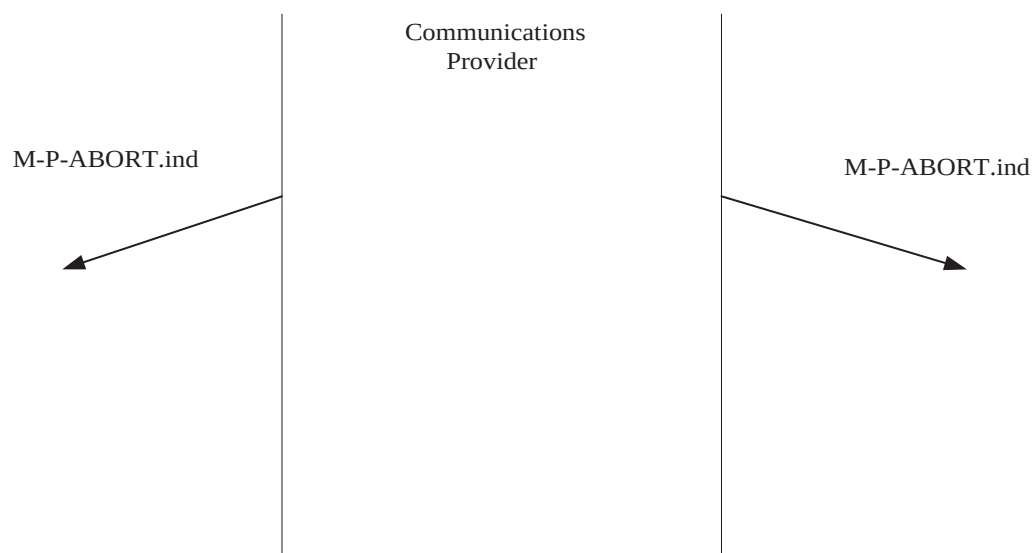


Figure 6 - M-P-ABORT service

Table 5 shows the parameters of the M-P-ABORT service.

Table 5 - M-P-Abort service parameters

Parameters	Req	Ind
Abort Source		M

6.9.5.1 Abort Source

This parameter is present in the indication and indicates the source of the abort request. If true, the abort was requested from the local system; if false, the abort was requested from the remote system.

7 The Virtual Manufacturing Device

7.1 Introduction

This clause provides object models for the following objects:

VMD
TRANSACTION

7.1.1 The VMD and the real manufacturing device

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 provides a model for a VMD. It explains the VMD's relationship to the underlying communication system. It defines the structural elements of the VMD and introduces the abstract objects that exist at a VMD and are manipulated on behalf of a MMS client.

An implementation of an MMS server provides 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 provided.

7.1.2 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 that 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 that 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. If a VMD contains more than one AE, it contains more than one set of communication capabilities.

7.1.3 Relationship of the VMD to a 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 that 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 16.)

MMS describes the operation of a VMD by describing the abstract objects that are manipulated by it, and by describing the set of operations that 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 using an ASN.1 module containing Object Class definitions for each object. The module begins with an ASN.1 module header exporting symbols to the module contained in ISO 9506-2 that defines the abstract syntax.

```
MMS-Object-Module-1 { iso standard 9506 part(1) mms-object-model-version1(2) }  
DEFINITIONS ::= BEGIN
```

```
--  
-- This ASN.1 specification has been checked for conformance with the  
-- ASN.1 standard by the OSS ASN.1 Tools.  
--
```

```

EXPORTS AccessCondition,
AdditionalCBBOptions,
AdditionalSupportOptions,
Address,
AlarmAckRule,
Control-State,
DomainState,
EC-State,
EC-Class,
EE-Duration,
EE-Class,
EventTime,
Journal-Variable,
LogicalStatus,
Modifier,
normalPriority,
normalSeverity,
ParameterSupportOptions,
PhysicalStatus,
Priority,
ProgramInvocationState,
Running-Mode,
ServiceSupportOptions,
Severity,
Transitions,
TypeDescription,
ULState,
VMDState;

```

```

IMPORTS ApplicationReference,
Authentication-value FROM
MMS-Environment-1 { iso standard 9506 part(2) mms-environment-version1(4) }
AlternateAccess,
ConfirmedServiceRequest,
AttachToSemaphore,
AttachToEventCondition,
Data,
EE-State,
Identifier,
Integer8,
Integer32,
MMSString,
MMS255String,
ObjectName,
TimeOfDay,
TypeSpecification,
Unsigned32,
Unsigned8 FROM
ISO-9506-MMS-1 { iso standard 9506 part(2) mms-abstract-syntax-version1(1) };

```

-- Part 1 - Object Model Definitions

```

--
-- Note - ASN.1 rules for interpreting the object formalism.
--
-- Each field has a field identifier possibly followed by a name.
-- The field identifier begins with an '&' and is followed by a reference name,
-- beginning with either a lower case or an upper case letter.
--
-- If the field identifier begins with '&' Upper case letter:
--
-- If there is no following name,
-- the field identifies a type.
-- If the following name is mixed case
-- and begins with an upper case letter,
-- or if the following name is upper case
-- and the name of a Universal type,
-- the field identifies a value set.
-- If the following name is upper case
-- and the name of an Object Class,
-- the field identifies an Object Set.
--
-- If the field identifier begins with '&' lower case letter:
--
-- If the following name is upper case
-- and the name of an Object Class,
-- the field identifies an Object (instance).
-- If the following name is mixed case
-- and begins with an upper case letter,
-- or if the following name is upper case
-- and is the name of a Universal type,
-- the field identifies a value.
--

```

7.2.1 VMD Object Model

This clause introduces the model for the VMD.

```

VMD ::= CLASS {
    &executiveFunction      ApplicationReference,
    &vendorName             MMSString,
    &modelName              MMSString,
    &revision               MMSString,
    &AbstractSyntaxes       ABSTRACT-SYNTAX OPTIONAL,
    &EATransactions         TRANSACTION OPTIONAL,
    &Associations           APPLICATION-ASSOCIATION,
    &accessControl          Identifier,
    &logicalStatus          LogicalStatus,
    &Capabilities           MMSString,
    &physicalStatus         PhysicalStatus,
    &local-detail           BIT STRING(SIZE (1..128)),
    &AccessControllists     ACCESS-CONTROL-LIST OPTIONAL,
    &Domains                DOMAIN OPTIONAL,
    &ProgramInvocations     PROGRAM-INVOCATION OPTIONAL,
    &UnitControls           UNIT-CONTROL OPTIONAL,
    IF (vadr)
    &UnnamedVariables       UNNAMED-VARIABLE OPTIONAL,
ENDIF
    IF (vnam)
    &NamedVariables         NAMED-VARIABLE OPTIONAL,
    IF (vlis)
    &NamedVariableLists     NAMED-VARIABLE-LIST OPTIONAL,
ENDIF
    &NamedTypes            NAMED-TYPE OPTIONAL,
ENDIF
    &DataExchanges         DATA-EXCHANGE OPTIONAL,
    &Semaphores            SEMAPHORE OPTIONAL,
    &OperatorStations      OPERATOR-STATION OPTIONAL,
    &EventConditions       EVENT-CONDITION OPTIONAL,
    &EventActions          EVENT-ACTION OPTIONAL,
    &EventEnrollments      EVENT-ENROLLMENT OPTIONAL,
    IF (cspi)
    &EventConditionLists    EVENT-CONDITION-LIST OPTIONAL,
ENDIF
    &Journals              JOURNAL OPTIONAL
    IF (csr)
    , &operationState       VMDState,
    &safety-Interlocks-Violated BOOLEAN,
    &any-Resource-Power-On  BOOLEAN,
    &all-Resources-Calibrated BOOLEAN,
    &local-Control          BOOLEAN,
    &selected-Program-Invocation Identifier
ENDIF
}

```

7.2.1.1 &executiveFunction

The &executiveFunction field identifies the VMD. The value of this field is of type ApplicationReference. For OSI specific communications systems, this type is defined in Annex A. For other communication systems, a similar definition of ApplicationReference must be supplied. The existence of a fully functional &executiveFunction exactly corresponds with the existence of the VMD.

7.2.1.2 &vendorName

The &vendorName field is a character string that identifies the vendor of the system that supports this VMD.

7.2.1.3 &modelName

The &modelName field is a character string that identifies the model of the system that supports this VMD. The value of this string is normally assigned by the vendor.

7.2.1.4 &revision

The &revision field is a character string that identifies the revision level of the system that supports this VMD. The value of this string is normally assigned by the vendor.

7.2.1.5 &AbstractSyntaxes

The &AbstractSyntaxes field identifies the set of Abstract Syntaxes that this VMD is able to support in the MMS Application Context. This field includes any abstract syntax that can be recognized as an encoding for Load Data and Execution Argument. The abstract syntax defined in ISO 9506-2 shall not be included in the field. The ABSTRACT-SYNTAX object is defined in ISO/IEC 8824-2, Annex B.

7.2.1.6 &EATransactions

This field identifies those transaction objects that do not explicitly depend on an application association. Such transactions occur through the processing of Event Actions. Transaction objects are described in 7.3.1.

7.2.1.7 &Associations

This field identifies the associations established between this VMD and external MMS clients. In order for MMS services to be used, at least one such association is required. Association establishment and the Application Association object are described in clause 8.

7.2.1.8 &accessControl

This field identifies an Access Control List object that specifies necessary (but not sufficient) conditions for an MMS service to succeed. The conditions specified in this Access Control List object shall be satisfied for the service class corresponding to the requested service in order for the service to succeed. Additional conditions for success may be imposed by an Access Control List object referenced by the object of the service request. If no other specification has been provided, this field should reference the predefined symbol 'M_NonDeletable'. The Access Control List object is described in clause 9.

7.2.1.9 &logicalStatus

The &logicalStatus field identifies one of four levels of functionality of the VMD available through MMS.

```
LogicalStatus ::= [0] IMPLICIT INTEGER {
    state-changes-allowed      (0),
    no-state-changes-allowed   (1),
    limited-services-permitted  (2),
    support-services-allowed    (3) } (0..3)
```

7.2.1.9.1 state-changes-allowed

If &logicalStatus has this value, all MMS services supported by this VMD may be performed.

7.2.1.9.2 no-state-changes-allowed

If &logicalStatus has this value, the only MMS services that may be performed (if the VMD supports the service) are:

Abort	GetUnitControlAttributes
Conclude	GetVariableAccessAttributes
Cancel	Identify
GetAccessControlListAttributes	Initiate
GetAlarmEnrollmentSummary	Read
GetAlarmSummary	ReadJournal
GetCapabilityList	ReportAccessControlledObjects
GetDataExchangeAttributes	ReportECLStatus
GetDomainAttributes	ReportEventActionStatus
GetECLAttributes	ReportEventConditionStatus
GetEventActionAttributes	ReportEventEnrollmentStatus
GetEventConditionAttributes	ReportJournalStatus
GetEventEnrollmentAttributes	ReportPoolSemaphoreStatus
GetNamedTypeAttributes	ReportSemaphoreEntryStatus
GetNamedVariableListAttributes	ReportSemaphoreStatus
GetNameList	Status
GetProgramInvocationAttributes	

7.2.1.9.3 limited-services-permitted

If &logicalStatus has this value, the only MMS services that may be performed are: Abort, Conclude, Status and Identify.

7.2.1.9.4 support services allowed

If &logicalStatus has this value, all MMS services supported by the VMD except the Start, Stop, Reset, Resume, Kill, StartUnitControl, and StopUnitControl services may be performed.

7.2.1.10 &Capabilities

A VMD represents the real device to the MMS client by providing capabilities that may be used by the MMS 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 that can be identified by a character string. The definition and management of capabilities is outside the scope of this part of ISO 9506. However, it is assumed that the existence and status of capabilities are known by the &executiveFunction, and that the &executiveFunction contains sufficient knowledge in order to "allocate" capabilities to the various Domains that 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 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 field "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.

7.2.1.11 &physicalStatus

Associated with each capability is one or more attributes that describe the state of the capability. It is outside the scope of this part of ISO 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 &physicalStatus field represents this attribute (of the entire collection of capabilities) 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.

```
PhysicalStatus ::= [1] IMPLICIT INTEGER {
    operational          (0),
    partially-operational (1),
    inoperable           (2),
    needs-commissioning  (3) } (0..3)
```

7.2.1.11.1 operational

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

7.2.1.11.2 partially-operational

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

7.2.1.11.3 inoperable

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

7.2.1.11.4 needs-commissioning

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

7.2.1.12 &local-detail

This attribute is a bitstring that represents additional detail about the status of the VMD as specified by the vendor of the device. The maximum length of the bitstring is 128 bits. The contents of this attribute is a local matter and not a subject for further standardization.

7.2.1.13 &AccessControlLists

This field identifies the Access Control List objects whose name scope is VMD-specific. Access Control List objects are described in clause 9.

7.2.1.14 &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 that are associated with specific elements (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, the Domain is predefined within the MMS server and its name is 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 is provided in clause 11.

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 that 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) that is part of the Application Process that 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.

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

7.2.1.15 &ProgramInvocations

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 12.

7.2.1.16 &UnitControls

This field identifies the Unit Control objects in the VMD. Unit Controls always have a name scope that is VMD-specific. Unit Control objects are described in clause 13.

7.2.1.17 &UnnamedVariables

The VMD may allow direct access to the memory space of the underlying computer. This field is present only if the **vadr** parameter CBB has been negotiated. Such access is characterized by a machine address and possibly a type specification corresponding to that address. Unnamed Variables are described in clause 14.

7.2.1.18 &NamedVariables

This field identifies the Named Variable objects whose name scope is VMD-specific. This field is present only if the **vnam** parameter CBB has been negotiated. Named Variables are described in clause 14.

7.2.1.19 &NamedVariableLists

This field identifies the Named Variable List objects whose name scope is VMD-specific. This field is present only if the **vnam** and the **vlis** parameter CBB's have been negotiated. Named Variables Lists are described in clause 14.

7.2.1.20 &NamedTypes

This field identifies the Named Type objects whose name scope is VMD-specific. This field is present only if the **vnam** parameter CBB has been negotiated. Named Types are described in clause 14.

7.2.1.21 &DataExchanges

This field identifies the Data Exchange objects whose name scope is VMD-specific. Data Exchange objects are described in clause 15

7.2.1.22 &Semaphores

This field identifies the Semaphore objects whose name scope is VMD-specific. Semaphores are described in clause 16.

7.2.1.23 &OperatorStations

This field identifies the Operator Station objects whose name scope is VMD-specific. Operator Stations are described in clause 17.

7.2.1.24 &EventConditions

This field identifies the Event Condition objects whose name scope is VMD-specific. Event Conditions are described in clause 19 and event processing is described in clause 18.

7.2.1.25 &EventActions

This field identifies the Event Action objects whose name scope is VMD-specific. Event Actions are described in clause 20 and event processing is described in clause 18.

7.2.1.26 &EventEnrollments

This field identifies the Event Enrollment objects whose name scope is VMD-specific. Event Enrollments are described in clause 21 and event processing is described in clause 18.

7.2.1.27 &EventConditionLists

This field identifies the Event Condition List objects whose name scope is VMD-specific. Event Condition Lists are described in clause 22 and event processing is described in clause 18.

7.2.1.28 &Journals

This field identifies the Journal objects whose name scope is VMD-specific. Journals are described in clause 16.

7.2.1.29 &operationState

This field represents the state of the system. This field is present only if the **csr** parameter CBB has been negotiated. The field is used solely for systems (such as robots) in which the VMD represents a single, complex piece of machinery with operational states. The relationship of this field to the physical devices and to the state of the Controlling Program Invocations is defined below.

```

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

```

7.2.1.30 &safety-Interlocks-Violated

This field, of type boolean, indicates the state of the safety interlocks. This field is present only if the **csr** parameter CBB has been negotiated. The field is used solely for robots and similar equipment having a safety interlock mechanism. If the value of this field is true, the safety interlocks have been violated since the system was last reset. The method of resetting the safety interlocks is a local matter.

7.2.1.31 &any-Resource-Power-On

This field, of type boolean, indicates whether (true) or not (false) any physical resource has power applied to it. This field is present only if the **csr** parameter CBB has been negotiated. The value of this field is the logical 'OR' of a similar primitive attribute of each physical resource in the system.

7.2.1.32 &all-Resources-Calibrated

This field, of type boolean, indicates whether (true) or not (false) all the physical resources in the system that have calibration attributes are in fact calibrated. This field is present only if the **csr** parameter CBB has been negotiated.

7.2.1.33 &local-Control

This field, of type boolean, indicates whether (true) or not (false) some local agent has control of any of the physical resources of the system. This field is present only if the **csr** parameter CBB has been negotiated. Having control indicates the ability to cause actions that change the physical resources and hence the attributes that represent those resources. &local-Control can reflect either a human operator or some automatic procedure, either of which inhibit the assertion of control remotely.

If &local-Control is false, control may reside in some remote agent. This condition can also reflect a condition in which the system is running under no direct control other than its task program.

The value of the &local-Control field, the &operation-State field, and the &logical-Status field are interrelated. Table 6 illustrates the relationship between these attributes.

Table 6 - Local control

Local Control	Operation State	VMD Logical Status
TRUE	Any	NO-STATE-CHANGES-ALLOWED or LIMITED-SERVICES-PERMITTED or SUPPORT-SERVICES-ALLOWED
FALSE	MANUAL-INTERVENTION-REQUIRED	NO-STATE-CHANGES-ALLOWED or LIMITED-SERVICES-PERMITTED or SUPPORT-SERVICES-ALLOWED
	Any other	Any

NOTE While &local-Control is true, the &operation-State can continue to change. However, it is a local matter how the state changes occur.

7.2.1.34 &selected-Program-Invocation

This field identifies the Program Invocation that has been selected to control the operation of the system. This field is present only if the **csr** parameter CBB has been negotiated. This Program Invocation has its &control field equal to **controlling** and has been selected through the use of the Select operation.

7.2.2 Initialization of the VMD

Clause 25 of ISO 9506-2 describes the initialization of the VMD.

7.2.3 Services on the VMD

Clause 10 describes the MMS services that operate on the VMD in its entirety.

7.3 Transactions

This clause introduces a model of the Transaction and its processing.

7.3.1 Transaction Object Model

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

```
TRANSACTION ::= CLASS {
    &invokeID                      INTEGER UNIQUE,
    IF ( attachToEventCondition attachToSemaphore )
        &Pre-executionModifiers  ModifierStep OPTIONAL,
        &currentModifier          CHOICE {
            modifier              ModifierStep,
            none                   NULL },
    ENDIF
    &confirmedService-Request     ConfirmedServiceRequest,
    IF ( attachToSemaphore )
        &Post-executionModifiers  ModifierStep OPTIONAL,
    ENDIF
    &cancelable                   BOOLEAN }

```

7.3.1.1 &invokeID

This field is an integer that serves to identify the transaction within the Application Association.

7.3.1.2 &Pre-executionModifiers

This field is an ordered set of modifiers to be satisfied before execution of the Confirmed Service Request can begin. This set may be empty.

```
ModifierStep ::= SEQUENCE {
    modifierID    INTEGER,
    modifier      Modifier
}

Modifier ::= CHOICE {
    IF ( attachToEventCondition )
        eventModifier      [0] IMPLICIT AttachToEventCondition
    IF ( attachToSemaphore )
        semaphoreModifier   [1] IMPLICIT AttachToSemaphore
    /
    ENDIF
    ENDIF
    IF ( attachToSemaphore )
        semaphoreModifier   [1] IMPLICIT AttachToSemaphore
    ENDIF
}

```

7.3.1.2.1 eventModifier

The AttachToEventCondition modifier is described in clause 18.

7.3.1.2.2 semaphoreModifier

The AttachToSemaphore modifier is described in clause 16.

7.3.1.3 ¤tModifier

This field identifies the Semaphore Entry object or the Event Enrollment object controlling the current modifier's execution. This field is initially empty. It changes to identify the modifier in the Set of Pre-Execution Modifiers as they are processed. If the Set of Pre-Execution Modifiers is empty, this field will always be **none**.

7.3.1.4 &ConfirmedService-Request

This field identifies the pending service, including its arguments.

7.3.1.5 &Post-executionModifiers

This field identifies the set of Semaphore Entry objects that this service invocation owns due to a processed Attach to Semaphore modifier. Attach to Event Condition modifiers do not occur on the list of &Post-executionModifiers.

7.3.1.6 &cancelable

This field identifies whether or not the service may be cancelled. Initially true, the MMS server may set this field false during the processing of the service request.

7.3.2 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 that may exist at any time is governed by the negotiated maximum number of services outstanding (see 8.2).

Upon receipt of a Confirmed Service indication, the MMS Server shall create a new Transaction object and initialize it as follows:

```
transaction TRANSACTION ::= {
  &invokeID                newID,
  &Pre-executionModifiers  serviceParameter,
  &CurrentModifier          none: NULL,
  &confirmedService-Request serviceParameter,
  &Post-executionModifiers { },
  &cancelable              TRUE
}
```

NOTE This production is not valid ASN.1 as written. The MMS Server should substitute a valid integer value for newID to identify the transaction, and insert the parameters from the Confirmed Service indication for the &Pre-executionModifiers and the &confirmedService-Request.

7.3.2.1 &invokeID

The MMS server shall assign an integer value to this field. This value shall be unique to this application association.

7.3.2.2 &Pre-executionModifiers

The MMS server shall assign a value set to this field corresponding to the parameters of the service indication.

7.3.2.3 &CurrentModifier

The MMS server shall assign an initial value of **none** to this field.

7.3.2.4 &confirmedService-Request

The MMS server shall assign a value to this field corresponding to the parameter of the service indication.

7.3.2.5 &Post-executionModifiers

The MMS server shall assign an empty value set as the initial value of this field.

7.3.2.6 &cancelable

The MMS server shall assign the value **TRUE** as the initial value of this field. 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.

7.3.3 Processing of Transaction objects

A Transaction object shall be created either upon receipt of an indication service primitive for an MMS confirmed service or as part of the processing of an event occurrence (see 18.1.1). The Transaction object shall be deleted after the MMS-user issues a response service primitive for that service instance. The number of Transaction objects that may exist at any time is governed by the negotiated maximum number of services outstanding (see 8.2).

NOTE Although this part of ISO 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 is processed. If a modifier fails (see definition of particular modifiers), the MMS-user shall process the Post-execution modifier list (as specified below) and issue a Response(-) service primitive, specifying the appropriate error class and code, and the Transaction object shall be deleted. The &CurrentModifier field shall be set to indicate the modifier currently being processed in the list. The &CurrentModifier field shall be identified by an integer, where 1 indicates the first element in the list. For each AttachToSemaphore modifier that is successfully processed, an entry shall be made in the &Post-executionModifiers field (in order to allow relinquishing of control of the semaphores after service execution). The &Post-executionModifiers field 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 9506 for the named service.

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

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

7.4 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.

7.4.1 Scope of Names

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

7.4.1.1 VMD-specific Scope

A name that 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 persist after the MMS application association ceases to exist.

7.4.1.2 Domain-specific Scope

A Domain represents a single flat name space. A name that 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 naming hierarchy.

7.4.1.3 Application Association-specific Scope

A name having "Application Association-specific" (AA-specific) scope may only be referenced by the MMS client 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.

7.4.2 Classes of objects

The specific instances of MMS objects that can be named are listed in Table 7. Not all objects can have all possible Name Scopes. The allowed combinations are indicated by Xs in Table 7.

Table 7 - Name Class and Scope

	VMD	Domain	AA	clause
Access Control List Objects	X			9
Domain Objects	X			11
Program Invocation Objects	X			12
Unit Control Objects	X			13
Named Variable Objects	X	X	X	14
Named Variable List Objects	X	X	X	14
Named Type Objects	X	X	X	14
Data Exchange Objects	X			15
Semaphore Objects	X			16
Operator Station Objects	X			17
Event Condition Objects	X	X	X	19
Event Action Objects	X	X	X	20
Event Enrollment Objects	X	X	X	21
Event Condition Lists	X	X	X	22
Journal Objects	X	X	X	23

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 16.4). Therefore, a semaphore cannot be proposed for creation that 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 12.2). Therefore, a Program Invocation cannot be proposed for creation whose name is the same as an existing Event Condition, Event Action, or Event Enrollment.

7.4.3 Object lifetime

Each of the objects listed above has a lifetime within the VMD that 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).

7.4.4 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 on the association over which the object was defined.

7.4.5 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 usually may not be deleted through the use of MMS services, and dynamic objects usually may be deleted, 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 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 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.

7.4.6 Deletion of MMS objects

All MMS objects subordinate to the VMD may be deleted from the VMD through appropriate MMS service requests if such requests are permitted (see 9.1.1.8). 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 9506. Objects that are subordinate to a Domain are automatically deleted when the Domain is deleted. This is true regardless of any conditions specified in the Access Control List object referenced by the object subordinate to the Domain.

7.4.7 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 that 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 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.

7.4.8 Control of Access to MMS objects

This International Standard provides explicit control for the ability to access or alter MMS named objects. Each named object within an MMS implementation contains a reference to an Access Control List object that specifies the conditions under which services directed at the named object may succeed. For the purposes of specifying the control conditions, services are grouped into seven classes, read, write, load, store, execute, delete, and edit. The control conditions include possession of a semaphore, identity of user (Application Reference), and the submission of a password (which may be arbitrarily complex). These conditions are necessary but not sufficient for the success of the service. If the conditions are not satisfied, the service is required to fail; the service may always fail for reasons beyond the scope of this standard. These conditions may be combined in arbitrary ways. Conditions may be specified separately for individual objects and for all objects of the VMD. Conditions restricting creation of objects can only be specified for the entire VMD.

In the second amendment to the first edition of ISO/IEC 9506:1990, the &accessControlList field of named objects replaced the MMS Deletable attribute of the first edition of MMS. For backward compatibility, a derivation rule

from the &accessControlList is provided for services that report the value of the MMS Deletable attribute. Using this rule, implementations of earlier versions of MMS will be able to interwork with implementations of this version of MMS as long as the additional services specified in this version are not employed.

The **aco** parameter CBB is used to indicate whether or not the object reporting services shall report attributes related to the use of Access Control Lists.

7.5 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 8.

Table 8 - 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(=)

7.5.1 Name Scope

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

7.5.2 AA-specific

Some named objects may have an AA-specific name. AA-specific objects shall only exist as a result of a definition of the name over the association by an MMS Service. As there is no MMS service to create AA-specific Data Exchange objects, these objects may be created by local action at the time of association establishment.

7.5.2.1 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.

7.5.3 Domain-specific

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

7.5.3.1 Domain Identifier

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

7.5.3.2 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.

7.5.4 VMD-specific

Any named object may have a VMD-specific name.

7.5.4.1 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.

7.6 Object Class structure

The parameter Object Class occurs frequently in the specification of MMS services. The structure of the Object Class is shown in Table 9.

Table 9 - Object Class

Object Class	Req/Rsp	Ind/Cnf
Object Class	M	M(=)
Basic Object Class	S	S(=)
Extended Object Class	S	S(=)

7.6.1 Object Class

This parameter specifies the class of object under consideration. For historical reasons, the choices fall into one of categories. One of the following parameters shall be chosen.

7.6.2 Basic Object Class

This parameter shall be chosen if the object is one of the following:

- | | |
|------------------------|------------------------|
| a) Named Variable | h) Event Enrollment |
| b) Scattered Access | i) Journal |
| c) Named Variable List | j) Domain |
| d) Named Type | k) Program Invocation |
| e) Semaphore | l) Operator Station |
| f) Event Condition | m) Data Exchange |
| g) Event Action | n) Access Control List |

7.6.3 Extended Object Class

This parameter shall be chosen if the object is one of the following:

- a) EventConditionList
- b) UnitControl

8 Environment And General Management services

8.1 Introduction and Models

This clause provides a model for the following object:

APPLICATION-ASSOCIATION

This clause specifies the following services:

Initiate
Conclude
Abort

Cancel
Reject