



TECHNICAL REPORT

EIA-511™

Issued 1989-03
Stabilized 2015-07

Manufacturing Message Specification – Service Definition and Protocol

NOTICE

This document has been taken directly from the original TechAmerica document and contains only minor editorial and format changes required to bring it into conformance with the publishing requirements of SAE Technical Standards. The release of this document is intended to replace the original with the SAE International document. Any numbers established by the original document remain unchanged.

The original document was adopted as an SAE publication under the provisions of the SAE Technical Standards Board (TSB) Rules and Regulations (TSB 001) pertaining to accelerated adoption of specifications and standards. TSB rules provide for (a) the publication of portions of unrevised specifications and standards without consensus voting at the SAE committee level, and (b) the use of the existing specification or standard format.

SAE Technical Standards Board Rules provide that: "This report is published by SAE to advance the state of technical and engineering sciences. The use of this report is entirely voluntary, and its applicability and suitability for any particular use, including any patent infringement arising therefrom, is the sole responsibility of the user."

SAE reviews each technical report at least every five years at which time it may be revised, reaffirmed, stabilized, or cancelled. SAE invites your written comments and suggestions.

Copyright © 2016 SAE International

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of SAE.

TO PLACE A DOCUMENT ORDER: Tel: 877-606-7323 (inside USA and Canada)
Tel: +1 724-776-4970 (outside USA)
Fax: 724-776-0790
Email: CustomerService@sae.org
<http://www.sae.org>

SAE WEB ADDRESS:

Full document details: SAE EIA511

SAE values your input. To provide feedback on this Technical Report, please visit <http://www.sae.org/technical/standards/EIA511>

Documents

Document Sharing Hub

ERRATA

(The changes indicated below were made subsequent to the printing of the main body of EIA-511 "Manufacturing Message Specification - Service Definition and Protocol", to bring about consistency with ISO-9506.)

1.

Service-Page 10

Section 2

In the list of Normative References in both parts, there should be a mention of the Addendum to ISO 8824 and to ISO 8825.

ISO 8824;1987	Information Processing Systems Open Systems Interconnection - Specification of Abstract Syntax Notation One(ASN.1)
ISO 8824/AD 1	Information Processing Systems Open Systems Interconnection - Specification of Abstract Syntax Notation One(ASn.1) Addendum 1: ASN.1 Extensions
ISO 8825;1987	Information processing Systems Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One(ASN.1)
ISO 8825/AD 1	Information processing Systems Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One(ASN.1) Addendum 1: ASN.1 Extensions

2.

Service-Page 94

Section 10.3.1.2.1

In the Note at the bottom of the page, the word "regarding" is misspelled.

3.

Service-Page 162

Section 12.2.1.5.1

Following the last element of the list which begins 'k) BCD - ...', insert a new list element:

- 1) OBJECT IDENTIFIER - This type shall consist of the binary encoding of an Object Identifier value as given in ISO 8825. The Size parameter shall be omitted.

Note that the index is a lower case 'L', not the numeral 1.

4.

Service-Page 308

Section 15.12.1.3.1

In the fourth line, after "or the Acknowledgement Event Condition parameter", insert the following text:

", or the Event Action object specified by the Event Action Name parameter"

5.

Service-Page 308

Section 15.12.2

At the beginning of the paragraph, insert the following text:

"The MMS server shall perform the following actions:

- a) Verify that no Event Enrollment object exists with the same name as the Event Enrollment Name parameter;
- b) Verify the existence of an Event Condition object with the same name as the Event Condition Name parameter;
- c) Verify the existence of an Event Action object with the same name as the Event Action Name parameter, if present; and
- d) Verify the existence of an Event Condition object with the same name as the Acknowledgement Event Condition parameter, if present.

If an Event Condition object or an Event Action object do not exist, then the responding MMS-user shall issue a Define Event Enrollment service response primitive with the Result(-) parameter with the Error Class of ACCESS, Error Code of OBJECT-NON-EXISTENT, and Object Not Defined parameter containing the value of the name which did not exist."

In the first sentence of the present paragraph, replace the phrase "If the requested definition is acceptable, a notification..." with:

"A notification..."

6.

Protocol-Page 12

Section 2

In the list of Normative References in both parts, there should be a mention of the Addendum to ISO 8824 and to ISO 8825.

ISO 8824;1987	Information Processing Systems Open Systems Interconnection - Specification of Abstract Syntax Notation One(ASN.1)
ISO 8824/AD 1	Information Processing Systems Open Systems Interconnection - Specification of Abstract Syntax Notation One(ASN.1) Addendum 1: ASN.1 Extensions
ISO 8825;1987	Information processing Systems Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One(ASN.1)
ISO 8825/AD 1	Information processing Systems Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One(ASN.1) Addendum 1: ASN.1 Extensions

7.

Protocol-Page 34

Section 7

In the protocol, the EXPORTS statement must precede the IMPORTS statement.

8.

Protocol-Page 40

The GetNameList Service should be included in the CS-Request-Detail.

```
CS-Request-Detail ::= CHOICE {
    status           [0] IMPLICIT CS-Status-Request,
    getNameList      [1] IMPLICIT CS-GetNameList-Request,
    ...
```

9.

Protocol-Page 45

Section 7.5.4

The GetNameList Service should be included in the CS-Response-Detail.

```
CS-Response-Detail ::= CHOICE {
    status           [0] IMPLICIT CS-Status-Response,
    getNameList      [1] IMPLICIT CS-GetNameList-Response,
    ...
```

10.

Protocol-Page 49

Section 7.5.5

There needs to be a hook so that Companion Standards can extend the definition of ServiceError. Change the protocol to include:

```

...
others [12] IMPLICIT INTEGER, -- OTHERS
cs-error [13] CS-Service-Error
-- shall not be chosen in the abstract syntax defined
-- Clause 19 of this Part of this International Standard
},
...

```

11.

Protocol-Page 50

Section 7.5.5

Remove the IMPLICIT from tag item 8. It should read:

```

defineEventEnrollment-Error
    [8] DefineEventEnrollment-Error,

```

12.

Protocol-Page 51

Section 7.6.1

There is incorrect syntax in the use of Subtypes in the ASN.1 notation. The TimeOfDay production should be replaced with

```

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

```

13.

Protocol-Page 52

Section 7.6.2

The Identifier production requires parentheses around the alphabet and around the SIZE specifier.

```

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

```

14.

Protocol-Page 64

Section 9.2

In the Status-Response production the SIZE modifier must be enclosed in parentheses.

```

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

```

15.

Protocol-Page 86

Section 12.2.1

In the TypeSpecification production, add a line following the last line of the production to reference object identifiers. The last two lines should now read:

```

bcd          [13] IMPLICIT Unsigned8, -- BCD
objId        [15] IMPLICIT NULL
}

```

16.

Protocol-Page 89

Section 12.4.2

In the Data production, add a line following the last line of the production to reference object identifiers. The last two lines should now read:

```

booleanArray [14] IMPLICIT BIT STRING,
objId        [15] IMPLICIT OBJECT IDENTIFIER
}

```

17.

Protocol-Page 116

Section 15.8

In the DefineEventAction-Request ASN.1 specification, the Companion Standard parameter is missing. The specification should read:

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

18.

Protocol-Page 118

Section 15.10

In the GetEventActionAttributes-Response ASN.1 specification, the Companion Standard parameter is missing. The specification should read:

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

19.

Protocol-Page 125

Section 15.17

In the EventNotification production, the success choice under the event ActionResult part of the actionResult is incorrect. It should read:

```
eventActionResult CHOICE {
  success          [0] IMPLICIT SEQUENCE [
    ConfirmedServiceResponse,
    [79] CS-Response-Detail OPTIONAL
    -- shall not be transmitted if value is NULL
  ],
  failure          [1] IMPLICIT ServiceError
}
```

etc.

20.

Protocol-Page 125 Section 15.17

The third line of the Event Notification production should read:

severity [2] IMPLICIT Severity

21.

Protocol-Page 161 Section 19

Before the END of the ASN.1 module, add the production

CS-Service-Error ::= NULL

22.

Protocol-Page 161 Section 19

Before the END of the ASN.1 module, delete the production

AS-Filter ::= NULL

23.

Protocol-Page 162 Section A-1

In the IMPORTS statement within the dashed lines, the Object Identifier for the MMS-General-Module is incorrect. It should be the same as the Object Identifier on page 34-Protocol.

```
IMPORTS MMSpdu,  
  FROM MMS-General-Module  
  [iso standard 9506 part(2) mms-general-module-version(2)];
```

24.

Protocol-Page 166 Section A-1

At the end of the list, delete the type

AS-Filter

25.

Protocol-Page 166 Section A-1

At the end of the list, add the type

CS-Service-Error

AMERICAN NATIONAL
STANDARD

ANSI/EIA-511-1989
APPROVED: March 24, 1989

EIA STANDARD

Manufacturing Message Specification – Service Definition and Protocol

EIA-511

EIA-511

MARCH 1989

ELECTRONIC INDUSTRIES ASSOCIATION
ENGINEERING DEPARTMENT



Documents
Document Sharing Hub

NOTICE

EIA ENGINEERING STANDARDS AND PUBLICATIONS ARE DESIGNED TO SERVE THE PUBLIC INTEREST THROUGH ELIMINATING MISUNDERSTANDINGS BETWEEN MANUFACTURERS AND PURCHASERS, FACILITATING INTERCHANGEABILITY AND IMPROVEMENT OF PRODUCTS, AND ASSISTING THE PURCHASER IN SELECTING AND OBTAINING WITH MINIMUM DELAY THE PROPER PRODUCT FOR HIS PARTICULAR NEED. EXISTENCE OF SUCH STANDARDS AND PUBLICATIONS SHALL NOT IN ANY RESPECT PRECLUDE ANY MEMBER OR NON-MEMBER OF EIA FROM MANUFACTURING OR SELLING PRODUCTS NOT CONFORMING TO SUCH STANDARDS AND PUBLICATIONS, NOR SHALL THE EXISTENCE OF SUCH STANDARDS AND PUBLICATIONS PRECLUDE THEIR VOLUNTARY USE BY THOSE OTHER THAN EIA MEMBERS, WHETHER THE STANDARD IS TO BE USED EITHER DOMESTICALLY OR INTERNATIONALLY.

RECOMMENDED STANDARDS AND PUBLICATIONS ARE ADOPTED BY EIA WITHOUT REGARD TO WHETHER OR NOT THEIR ADOPTION MAY INVOLVE PATENTS ON ARTICLES, MATERIALS, OR PROCESSES. BY SUCH ACTION, EIA DOES NOT ASSUME ANY LIABILITY TO ANY PATENT OWNER, NOR DOES IT ASSUME ANY OBLIGATION WHATEVER TO PARTIES ADOPTING THE RECOMMENDED STANDARD OR PUBLICATION.

THIS EIA STANDARD WAS THE BASIS FOR INTERNATIONAL STANDARDIZATION BY ISO TC184 IN DEVELOPMENT OF ISO 9506 WHICH IS EXPECTED TO BE PUBLISHED WITHIN A YEAR OF THE DATE OF THIS STANDARD. IT IS BELIEVED THAT ISO 9506 WILL BE IDENTICAL OR NEARLY IDENTICAL TO THIS STANDARD.

THIS STANDARD DOES NOT PURPORT TO ADDRESS ALL SAFETY PROBLEMS ASSOCIATED WITH ITS USE OR ALL APPLICABLE REGULATORY REQUIREMENTS. IT IS THE RESPONSIBILITY OF THE USER OF THIS STANDARD TO ESTABLISH APPROPRIATE SAFETY AND HEALTH PRACTICES AND TO DETERMINE THE APPLICABILITY OF REGULATORY LIMITATIONS BEFORE ITS USE.

(FROM STANDARDS PROPOSAL NO. 1393-B, FORMULATED UNDER THE COGNIZANCE OF THE EIA IE-31 COMMITTEE ON NUMERICAL CONTROL SYSTEMS AND EQUIPMENT.)

COPYRIGHT 1989

PUBLISHED BY

ELECTRONIC INDUSTRIES ASSOCIATION
ENGINEERING DEPARTMENT
1722 EYE STREET, N.W.
WASHINGTON, D.C. 20006

PRICE: \$60.00

ALL RIGHTS RESERVED

PRINTED IN U.S.A.

©EIA-511

PART I:
Manufacturing Message Specification
- Service Definition

1 Foreword

This Standard provides a wide variety of services useful for various manufacturing and process control devices. It is designed to be used both by itself and in conjunction with Companion Standards, which describe the application of subsets of these services to particular device types. Annexes A and B of parts 1 and 2 of this Standard are normative. Annex C of part 1 of this Standard and annex C of part 2 this Standard are informative only.

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.

2 Security Considerations

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

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

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

3 Complexity of Services and Requirements

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

CONTENTS

1	Foreword	1
2	Security Considerations	1
3	Complexity of Services and Requirements	1
0	INTRODUCTION	
1	SCOPE	
2	NORMATIVE REFERENCES	
3	DEFINITIONS	
3.1	Reference Model definitions	12
3.2	Service Convention definitions	12
3.3	Abstract Syntax Notation definitions	13
3.4	Other definitions	13
4	ABBREVIATIONS	
5	CONVENTIONS	
5.1	Base of Numeric Values	17
5.2	Service Parameter Description	17
5.3	Invocation Identifier on Service Primitives	19
5.4	List Of Modifier on Service Primitives	19
5.5	Addressing in MMS	19
5.6	Service Conventions	19
5.7	Calling and Called MMS-user	20
5.8	Sending and Receiving MMS-user and MPPM	20
5.9	Requesting and Responding MMS-user	20
5.10	Client and Server of a Service	21
5.11	Object Modeling	23
5.12	References to Objects	24
5.13	Parameter Types	24
6	MMS IN THE OSI ENVIRONMENT	
6.1	Information Processing Tasks and Real Systems	26
6.2	Application Processes	26
6.3	Interaction of Application Processes	26
6.4	Interaction of Application Processes in OSI	27
6.5	Structure of Application Entities	27
6.6	Addressing of Application Entities	27
6.7	Application Context	28
6.8	Presentation Context, Abstract Syntaxes, and Transfer Syntaxes	28
7	The VIRTUAL MANUFACTURING DEVICE	
7.1	Introduction	30

7.2	The Structure of a VMD	32
7.3	Specification of Named Objects	40
7.4	Object Name Structure	44
7.5	Services on the VMD	45
8	ENVIRONMENT AND GENERAL MANAGEMENT SERVICES	
8.1	Introduction	46
8.2	Initiate Service	49
8.3	Conclude Service	56
8.4	Abort Service	59
8.5	Cancel Service	61
8.6	Reject Service	63
9	VMD SUPPORT SERVICES	
9.1	Introduction	69
9.2	Status Service	70
9.3	UnsolicitedStatus Service	72
9.4	GetNameList Service	73
9.5	Identify Service	76
9.6	Rename Service	78
9.7	GetCapabilityList Service	81
10	DOMAIN MANAGEMENT SERVICES	
10.1	The Domain Object	83
10.2	InitiateDownloadSequence service	92
10.3	DownloadSegment Service	94
10.4	TerminateDownloadSequence Service	96
10.5	InitiateUploadSequence Service	98
10.6	UploadSegment Service	100
10.7	TerminateUploadSequence Service	102
10.8	RequestDomainDownload Service	104
10.9	RequestDomainUpload Service	106
10.10	LoadDomainContent Service	108
10.11	StoreDomainContent Service	111
10.12	DeleteDomain Service	114
10.13	GetDomainAttributes Service	116
11	PROGRAM INVOCATION MANAGEMENT SERVICES	
11.1	The Program Invocation Object	118
11.2	CreateProgramInvocation Service	127
11.3	DeleteProgramInvocation Service	131
11.4	Start Service	133
11.5	Stop Service	135
11.6	Resume Service	137
11.7	Reset Service	139
11.8	Kill Service	141
11.9	GetProgramInvocationAttributes Service	142
12	VARIABLE ACCESS SERVICES	
12.1	The MMS Variable Access Model	144

12.2	Specification of Types	157
12.3	Specification of Alternate Access	162
12.4	Specification of Data Values	167
12.5	Specification of Access to Variables	170
12.6	Read Service	176
12.7	Write Service	178
12.8	InformationReport Service	180
12.9	GetVariableAccessAttributes Service	182
12.10	DefineNamedVariable Service	184
12.11	DefineScatteredAccess Service	187
12.12	GetScatteredAccessAttributes Service	190
12.13	DeleteVariableAccess Service	193
12.14	DefineNamedVariableList Service	196
12.15	GetNamedVariableListAttributes Service	199
12.16	DeleteNamedVariableList Service	202
12.17	DefineNamedType Service	205
12.18	GetNamedTypeAttributes Service	207
12.19	DeleteNamedType Service	209
12.20	Conformance	211
12.21	Guidance To Implementors	214
13	SEMAPHORE MANAGEMENT SERVICES	
13.1	The Semaphore Management Model	217
13.2	TakeControl Service	226
13.3	RelinquishControl Service	230
13.4	DefineSemaphore Service	232
13.5	DeleteSemaphore Service	234
13.6	ReportSemaphoreStatus Service	235
13.7	ReportPoolSemaphoreStatus Service	237
13.8	ReportSemaphoreEntryStatus Service	239
13.9	AttachToSemaphore Modifier	242
13.10	Conformance	246
14	OPERATOR COMMUNICATION SERVICES	
14.1	The Operator Communications Model	247
14.2	Input Service	251
14.3	Output Service	254
15	EVENT MANAGEMENT SERVICES	
15.1	The Event Management Model	256
15.2	DefineEventCondition Service	282
15.3	DeleteEventCondition Service	286
15.4	GetEventConditionAttributes Service	288
15.5	ReportEventConditionStatus Service	290
15.6	AlterEventConditionMonitoring Service	292
15.7	TriggerEvent Service	295
15.8	DefineEventAction Service	297
15.9	DeleteEventAction Service	299
15.10	GetEventActionAttributes Service	302
15.11	ReportEventActionStatus Service	304
15.12	DefineEventEnrollment Service	306
15.13	DeleteEventEnrollment Service	311
15.14	GetEventEnrollmentAttributes Service	315
15.15	ReportEventEnrollmentStatus Service	322

15.16	AlterEventEnrollment Service	324
15.17	EventNotification Service	328
15.18	AcknowledgeEventNotification Service	331
15.19	GetAlarmSummary Service	333
15.20	GetAlarmEnrollmentSummary Service	337
15.21	Attach To Event Condition Modifier	342
15.22	Event Management State Diagrams	345
15.23	Conformance Requirements Unique to Event Management	353
16	JOURNAL MANAGEMENT SERVICES	
16.1	The Journal Management Model	354
16.2	ReadJournal Service	359
16.3	WriteJournal Service	371
16.4	InitializeJournal Service	375
16.5	ReportJournalStatus Service	377
16.6	CreateJournal Service	379
16.7	DeleteJournal Service	381
16.8	Conformance Requirements Unique to Journals	382
17	ERRORS	
17.1	Error Type	384
17.2	Description of structure of generic error type:	384
17.3	Additional Code	391
17.4	Additional Detail	392
17.5	Modifier Position	392
18	MMS STANDARDIZED NAMES	
18.1	Introduction	393
18.2	Unique Name Assignment Mechanism	393
18.3	MMS Standardized Names	393
19	CONFORMANCE	
19.1	Introduction	395
19.2	Conformance Building Blocks (CBBs)	395
19.3	Static Conformance Requirements	398
19.4	Calling MMS-user Conformance Requirements	399
19.5	Called MMS-user Conformance Requirements	399
19.6	Server Conformance Requirements	400
19.7	Client Conformance Requirements	401
19.8	Parameter CBB Conformance Requirements	402
19.9	Dynamic Conformance	403
Annex A	REQUIREMENTS FOR COMPANION STANDARDS (NORMATIVE)	
A.0	Introduction	404
A.1	Scope	404
A.2	References	404
A.3	Requirements	404
A.4	Outline of an MMS Companion Standard	405

Annex B FILE ACCESS SERVICE (NORMATIVE)

B.1	ObtainFile service	420
-----	------------------------------	-----

Annex C FILE MANAGEMENT SERVICES (INFORMATIVE)

C.1	Introduction	422
C.2	The MMS File Model	422
C.3	FileOpen Service	426
C.4	FileRead Service	428
C.5	FileClose Service	430
C.6	FileRename Service	431
C.7	FileDelete Service	433
C.8	FileDirectory Service	434
C.9	File Attributes Parameter	438
C.10	Additional Specification for the Conclude Service	439
C.11	Additional Specification for the Abort Service . .	439

0

INTRODUCTION

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 document is part of a set of standards produced to facilitate the interconnection of information processing systems. It is positioned within the application layer of the Open Systems Interconnection Environment as an Application Service Element (ASE) with respect to other 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 standard is to define the Manufacturing Message Specification Protocol. It is most closely related to and lies within the field of application of the Manufacturing Message Specification Service Definition (part 1 of this Standard). It also uses and references the Association Control Service Element Definition (ISO 8649) and the Presentation

©EIA-511

INTRODUCTION

Page 0-2

as defined in ISO 8649 and ISO 8822, respectively.

This Standard 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.

The second part of this Standard specifies the protocol that supports the Manufacturing Message Specification.

1

SCOPE

The Manufacturing Message Specification is an application layer Standard designed to support messaging communications to and from programmable devices in a Computer Integrated Manufacturing (CIM) environment. This environment is referred to in this Standard as the manufacturing environment. This Standard 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 Standard 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 this Standard, 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 Standard 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 this Standard does not specify individual implementations or products, nor does it constrain the implementation of entities and interfaces within a computer system. This Standard does specify the externally visible functionality of implementations, and specifies conformance requirements for such functionality.

NORMATIVE REFERENCES

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

- | | |
|-------------------|---|
| ISO 646:1983, | Information Processing - ISO 7-bit Coded Character Set for Information Interchange. |
| ISO 7498-1:1984, | Information Processing Systems - Open Systems Interconnection - Basic Reference Model. |
| ISO 7498-2, | Information Processing Systems - Open Systems Interconnection - OSI Security. To be published. |
| ISO 7498-3, | Information Processing Systems - Open Systems Interconnection - Naming and Addressing. To be published. |
| ISO 8326:1987, | Information Processing Systems - Open Systems Interconnection - Basic Connection Oriented Session Service Definition. |
| ISO TR/8509:1987, | Information Processing Systems - Open Systems Interconnection - Service Conventions. |
| ISO 8571:1987, | Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management. |
| ISO 8649:1987, | Information Processing Systems - Open Systems Interconnection - Association Control Service Element - Service Definition. |
| ISO 8650:1987, | Information Processing Systems - Open Systems Interconnection - Association Control Service Element - Protocol Specification. |
| ISO 8822:1987, | Information Processing Systems - Open Systems Interconnection - Connection Oriented Presentation Service Definition. |
| ISO 8824:1987, | Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1). |

NORMATIVE REFERENCES

Page 2-2

- ISO 8825:1987, Information Processing Systems - Open Systems
Interconnection - Specification of Basic Encoding
Rules for Abstract Syntax Notation One (ASN.1).
- ISO 9040, Information Processing Systems - Open Systems
Interconnection - Virtual Terminal Service Definition.
To be published.
- ISO 9041, Information Processing Systems - Open Systems
Interconnection - Virtual Terminal Protocol Specification.
To be published.
- ISO 9506-2:1988, Manufacturing Message Specification - Protocol
Specification.
- ISO 9545, Information Processing Systems - Open Systems
Interconnection - Application Layer Structure.
To be published.
- ISO 9594, Information Processing Systems - Open Systems
Interconnection - The OSI Directory Specification.
To be published.
- IEEE 754:1985, IEEE Standard for Binary Floating-Point Arithmetic.

3

DEFINITIONS

NOTE

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

For the purposes of this Standard, the definitions in the following subclauses apply.

3.1 Reference Model definitions

This Standard 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 Standard:

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

3.2 Service Convention definitions

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

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

DEFINITIONS

Abstract Syntax Notation definitions

Page 3-2

3.3 Abstract Syntax Notation definitions

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

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

3.4 Other definitions

For the purpose of this Standard, the following definitions also apply:

- 3.4.1 AA-specific (Application Association specific): An adjective used to describe an object whose name has a scope that is a single application association (i.e. the name may be referenced only on the application association with respect to which the object was defined).
- 3.4.2 attribute: A data element, having a defined meaning, together with a statement of the set of possible values it may take.
- 3.4.3 conformance building block (CBB): An atomic unit used to describe MMS conformance requirements.
- 3.4.4 Called MMS-user: The MMS-user that issues the Initiate.response service primitive.
- 3.4.5 Calling MMS-user: The MMS-user that issues the Initiate.request service primitive.
- 3.4.6 Client: The peer communicating entity which makes use of the VMD for some particular purpose via a service request instance.
- 3.4.7 data: Any representation to which meaning is or might be

DEFINITIONS

Other definitions

assigned (e.g. characters).

- 3.4.8 domain: An abstract object that represents a subset of the capabilities of a VMD which is used for a specific purpose.
- 3.4.9 Domain-specific: An adjective used to describe an object whose name has a scope that is a single domain (i.e. the name can be referenced over all application associations established with the VMD that may reference this domain).
- 3.4.10 download: The process of transferring the content of a domain, including any subordinate objects, via load data to an MMS-user.
- 3.4.11 event management: The management of event conditions, event actions, and event enrollments.
- 3.4.12 file: An unambiguously named collection of information having a common set of attributes.
- 3.4.13 file operation: The transfer of files between open systems, the inspection, modification or replacement of part of a file's content, or the management of a file and its attributes.
- 3.4.14 filestore: An organized collection of files, including their attributes and names, residing at a particular open system.
- 3.4.15 information: The combination of data and the meaning that it conveys.
- 3.4.16 journal: A set of recorded, time-tagged event transitions, variable data, and/or comments, which may be logically ordered during retrieval.
- 3.4.17 local matter: A decision made by a system concerning its behaviour in the Manufacturing Message Specification that is not subject to the requirements of this Standard.
- 3.4.18 Manufacturing Message Protocol Machine (MMPM): An abstract machine that carries out the procedures specified in this part of this Standard.
- 3.4.19 MMS-environment: A specification of the service elements of MMS and semantics of communication to be used during the lifetime of an application association.
- 3.4.20 MMS-provider: That part of the application entity that conceptually provides the MMS service through the exchange of MMS PDUs.
- 3.4.21 MMS-user: That portion of the application process which conceptually invokes the Manufacturing Message Specification.
- 3.4.22 monitored event: A detected change in the state of an event condition.
- 3.4.23 network-triggered event: An event which occurs due to an explicit solicitation by a client.
- 3.4.24 operator station: An abstract object representing equipment

DEFINITIONS
Other definitions

Page 3-4

associated with a VMD that provides for input/output interaction with an operator.

- 3.4.25 predefined object: An object, whose name is of VMD-specific, Domain-specific or Application Association-specific scope, that is instantiated through the use of some mechanism other than an MMS service.
- 3.4.26 program invocation: An abstract object representing a dynamic element which most closely corresponds to an execution thread in a multi-tasking environment, which is composed of a set of domains.
- 3.4.27 Receiving MMPM: The MMPM that receives an MMS PDU.
- 3.4.28 Receiving MMS-user: The MMS-user that receives an indication or confirmation service primitive.
- 3.4.29 remote device control and monitoring: The manipulation or inspection of the state of a device attached to the responder of a service request.
- 3.4.30 semaphore: A conceptual lock associated with a logical or physical resource that permits access to that resource only by an owner of the lock.
- 3.4.31 semaphore management: The control of semaphores.
- 3.4.32 Server: The peer communicating entity which behaves as a VMD for a particular service request instance.
- 3.4.33 Sending MMPM: The MMPM that sends an MMS PDU.
- 3.4.34 Sending MMS-user: The MMS-user issues a request or response service primitive.
- 3.4.35 standardized object: An object instantiation, whose name is of VMD-specific or Domain-specific scope, whose definition is provided in this Standard or an MMS Companion Standard.
- 3.4.36 type: An abstract description of a set of values which may be conveyed by the value of a variable.
- 3.4.37 upload: The process of transferring the content of a domain, including any subordinate objects, via load data from a remote user, in such a manner as to allow subsequent download.
- 3.4.38 variable: One or more data elements that are referred to together by a single name or description.
- 3.4.39 variable access: The inspection or modification of variables or components of variables defined at a VMD.
- 3.4.40 VMD-specific: An adjective used to describe an object whose name has a scope that is a single VMD (i.e. the name may be referenced by all application associations established with the VMD).

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
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 Standard uses a decimal representation for all numeric values unless otherwise noted.

5.2 Service Parameter Description

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

5.2.1 Companion Standard Service Parameters

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

5.2.2 Service Table Structure

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

- M - parameter is mandatory for the primitive
- U - parameter is a user option, and may or may not be provided depending on dynamic usage by the MMS-user
- C - parameter is conditional upon other parameters or the environment of the MMS-user
- (blank) parameter is never present
- COMP - parameter is for definition in MMS Companion Standards. Such parameters shall not be used other than as defined in a Companion Standard (and shall be omitted in the abstract syntax defined in this Standard). Annex C provides additional detail on the use of Companion Standards.

CONVENTIONS

Service Parameter Description

Page 5-2

S - parameter is a selection from a collection of two or more possible parameters. The parameters that make up this collection are indicated in the table as follows:

- a) each parameter in the collection is specified with the code "S";
- b) the name of each parameter in the collection is at the same indentation from the beginning of the parameter column in the table;
- c) Either
 - 1) each parameter is at the leftmost (outer) indentation in the table; or
 - 2) each parameter is part of the same parameter group. A parameter group is a collection of parameters where each group member has a common parent parameter. The parent parameter for any group member is the first parameter above the member that is not indented as far as that member. In the example below, ParameterA and ParameterB form a parameter group:

```

ParameterX
  ParameterA
  ParameterB
ParameterY
  ParameterC

```

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

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

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

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

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

CONVENTIONS

Service Parameter Description

Page 5-3

of this Standard.)

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

5.3 Invocation Identifier on Service Primitives

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

5.4 List Of Modifier on Service Primitives

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

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

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

5.5 Addressing in MMS

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

5.6 Service Conventions

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

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



Documents

Document Sharing Hub

CONVENTIONS
Service Conventions

Page 5-4

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

NOTE

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

5.7 Calling and Called MMS-user

This Standard makes use of the terms Calling and Called MMS-users. The Calling MMS-user is the MMS-user that issues the Initiate.request service primitive. The Called MMS-user is the MMS-user that issues the Initiate.response service primitive.

NOTE

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

5.8 Sending and Receiving MMS-user and MMPM

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

NOTE

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

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

5.9 Requesting and Responding MMS-user

This Standard 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

CONVENTIONS

Page 5-5

Requesting and Responding MMS-user

ACSE and other Standards. In those Standards, the term is used to reference the entity that responds to a connection request.

5.10 Client and Server of a Service

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

Figure 5-1 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 MPM.

©EIA-511

CONVENTIONS
Client and Server of a Service

Page 5-6

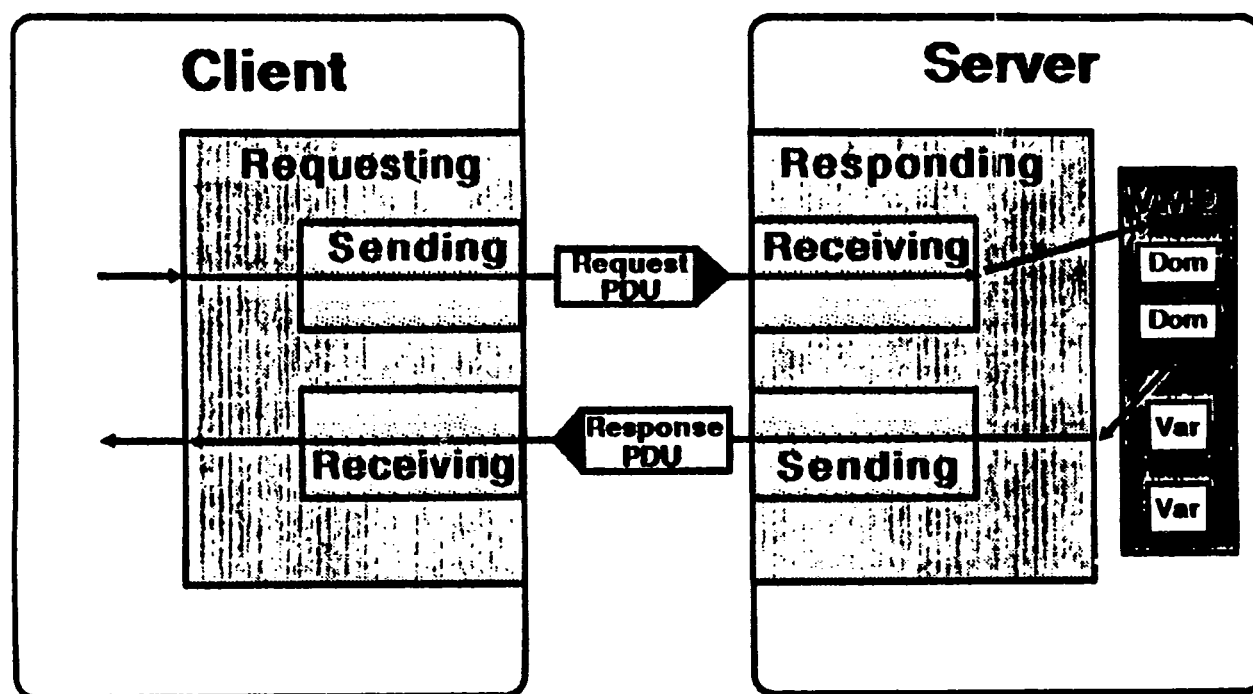


Figure 5-1: Relationships of Client and Server, Requesting and Responding MMS-user, and Sending and Receiving MPM

CONVENTIONS Object Modeling

Page 5-7

5.11 Object Modeling

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

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

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

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

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

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

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

By convention, each object definition begins with an object declaration and the name of the object. Immediately following, and indented, one or more key

Attributes are named. Next, zero or more Attributes are named. Note that constraints may be expressed anywhere within the attributes, with the convention that all attributes subject to the constraint are indented underneath it. The first attribute definition that is not indented ends the list of attributes that are subject to the constraint.

NOTE

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

5.12 References to Objects

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

NOTE

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

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

5.13 Parameter Types

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

6

MMS IN THE OSI ENVIRONMENT

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

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

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

NOTE 1

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

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

NOTE 2

This Standard specifies MMS operation in the OSI environment only. Operation in other environments and architectures is beyond the scope of this Standard, but may be standardized elsewhere.

6.1 Information Processing Tasks and Real Systems

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

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

6.2 Application Processes

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

6.3 Interaction of Application Processes

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

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

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

When two APs cooperate, their behaviour is determined partly by these models (the shared conceptual schema) and partly by their past interactions. The past integrations are modeled by the state of the objects in the universe of discourse. As an example, the cooperative behaviour of two APs may be partly determined by the Program Invocation model of MMS and partly by the state of

MMS IN THE OSI ENVIRONMENT

Interaction of Application Processes

Page 6-3

Program Invocation objects in the APs. The shared information about the state of objects is called an information base.

6.4 Interaction of Application Processes in OSI

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

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

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

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

NOTE

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

6.5 Structure of Application Entities

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

6.6 Addressing of Application Entities

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

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

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

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

NOTE

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

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

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

6.7 Application Context

The application context, which is established via the Association Control Service Element (ACSE) (ISO 8649), identifies the set of application service elements, their options, rules for use of the service elements, and their effects that are available on an application association. In the case of MMS, the application context associated with MMS indicates that the rules and requirements associated with this Standard 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. Part 2 of this Standard 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 part 2 of this Standard constitutes an abstract syntax. The MMS abstract syntax may be supported by many different transfer syntaxes.

MMS IN THE OSI ENVIRONMENT

Page 6-5

Presentation Context, Abstract Syntaxes, and Transfer Syntaxes

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

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

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

The VIRTUAL MANUFACTURING DEVICE

7.1 Introduction

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

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

NOTE

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

7.1.1 Relationship of the VMD to the OSI Model

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

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

EXAMPLE

An MMS system which is connected to a non-MMS environment containing multiple attached manufacturing devices could be modeled 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

The VIRTUAL MANUFACTURING DEVICE Introduction

Page 7-2

VMDs.

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

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

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

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

An OSI Presentation Address which identifies one or more PSAPs addresses a single VMD. This binding of a Presentation Address to a VMD is of a relatively long duration. Figure 7-1 illustrates the relationship between an OSI application process acting as an MMS server, its VMDs, and the PSAPs used to access them.

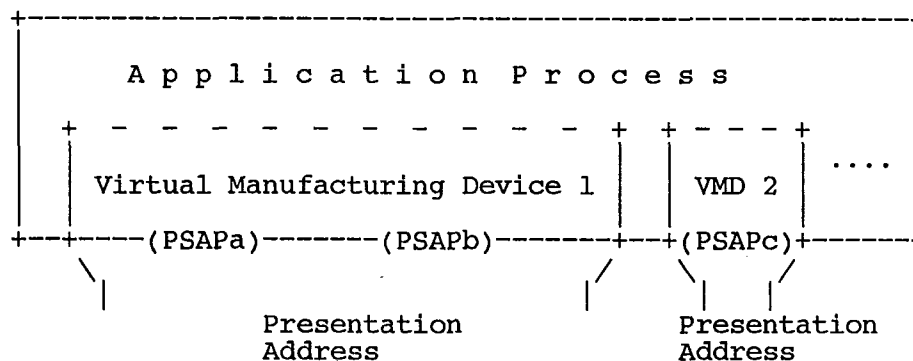


Figure 7-1: The MMS Server Application Process

7.1.2 Relationship of the VMD to the Real Manufacturing Device

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

The VIRTUAL MANUFACTURING DEVICE Introduction

Page 7-3

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

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

7.2 The Structure of a VMD

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

The VMD Object Model Object: VMD

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

7.2.1 The Executive Function

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

7.2.2 Vendor Name

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

The VIRTUAL MANUFACTURING DEVICE
The Structure of a VMD

Page 7-4

7.2.3 Model Name

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

7.2.4 Revision

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

7.2.5 List Of Abstract Syntaxes Supported

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

7.2.6 Logical Status

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

7.2.6.1 STATE-CHANGES-ALLOWED

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

7.2.6.2 NO-STATE-CHANGES-ALLOWED

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

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

©EIA-511

The VIRTUAL MANUFACTURING DEVICE
The Structure of a VMD

Page 7-5

ReportPoolSemaphoreStatus
ReportSemaphoreEntryStatus
ReportSemaphoreStatus
Status

7.2.6.3 LIMITED-SERVICES-PERMITTED

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

7.2.6.4 SUPPORT-SERVICES-ALLOWED

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

7.2.7 List Of Capabilities

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

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

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

NOTE

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

Example 1

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

The VIRTUAL MANUFACTURING DEVICE
The Structure of a VMD

Page 7-6

Example 2

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

7.2.8 Physical Status

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

NOTE

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

7.2.8.1 OPERATIONAL

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

7.2.8.2 PARTIALLY-OPERATIONAL

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

7.2.8.3 INOPERABLE

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

7.2.8.4 NEEDS-COMMISSIONING

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

7.2.9 List Of Program Invocations

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

©EIA-511

The VIRTUAL MANUFACTURING DEVICE The Structure of a VMD

Page 7-7

7.2.10 List Of Domains

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

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

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

EXAMPLE

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

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

7.2.11 Transaction Object

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

Object: Transaction

Key Attribute: Invoke ID
Key Attribute: Application Association Identifier
Attribute: List Of Pre-execution Modifiers
Attribute: Current Modifier Reference
Attribute: Confirmed Service Request



Documents
Document Sharing Hub

The VIRTUAL MANUFACTURING DEVICE
The Structure of a VMD

Page 7-8

Attribute: List Of Post-execution Modifiers
Attribute: Cancelable (TRUE, FALSE)

Invoke ID

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

Application Association Identifier

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

List Of Pre-execution Modifiers

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

Current Modifier Reference

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

Confirmed Service Request

Service identifier and Argument of the pending service.

List Of Post-execution Modifiers

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

Cancelable

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

7.2.11.1 Initialization of Transaction Objects

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

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

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

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



Documents

Document Sharing Hub

7.2.11.2 Processing of Transaction Objects

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

NOTE

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

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

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

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

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

7.2.12 Upload State Machines

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

7.2.13 Other VMD-specific Named Objects

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

The VIRTUAL MANUFACTURING DEVICE
The Structure of a VMD

Page 7-10

7.2.14 Additional Detail

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