

Fieldbus —

Part 6: Fieldbus Message Specification

ICS 25.040.40; 35.100

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW



Document Sharing Hub

Committees responsible for this Draft for Development

The preparation of this Draft for Development was entrusted to Technical Committee AMT/7, Monitoring and control aspects of advanced manufacturing technology, upon which the following bodies were represented:

- ABB Kent plc
- Cegelec Industrial Controls Ltd.
- Department of Trade and Industry
- Fisher-Rosemount Ltd.
- Foxboro Great Britain
- Institution of Electrical Engineers
- Measurement Technology Ltd.
- Process Automation and Computer Systems Ltd.
- Schneider Ltd.
- Siemens Automation plc

This Draft for Development, having been prepared under the direction of the Advanced Manufacturing Technology Assembly, was published under the authority of the Standards Board and comes into effect on 15 October 1996

© BSI 11-1998

The following BSI reference relates to the work on this Draft for Development:
Committee reference AMT/7

ISBN 0 580 26495 5

Amendments issued since publication

Amd. No.	Date	Comments

Contents

	Page
Committees responsible	Inside front cover
Foreword	vi
1 Scope	1
2 Architecture	1
3 FMS Models and Services	1
3.1 VFD Support	1
3.1.1 Model Description	1
3.1.2 The VFD Object	3
3.1.3 VFD Support Services	4
3.2 Object Dictionary (OD) Management	6
3.2.1 Model Description	6
3.2.2 Structure of an OD	7
3.2.3 Object Description in the OD	9
3.2.4 OD Object Description	9
3.2.5 The OD Object	11
3.2.6 OD Services	12
3.2.7 State Machine	16
3.3 Context Management	18
3.3.1 Model Description	18
3.3.2 The VCR List	18
3.3.3 The Transaction Object	21
3.3.4 Context Management Services	22
3.3.5 Tests at Connection Establishment	26
3.3.6 State Machine for VCRs	27
3.4 Domain Management	37
3.4.1 Model Description	37
3.4.2 The Domain Object	38
3.4.3 Generic Download Services	39
3.4.4 Download Services	42
3.4.5 Upload Services	44
3.4.6 State Machine for GenericDownload/Download	47
3.4.7 State Machine for Upload	50
3.4.8 Examples	51
3.5 Program Invocation Management	52
3.5.1 Model Description	52
3.5.2 The Program Invocation (PI) Object	52
3.5.3 Program Invocation Services	55
3.5.4 Program Invocation State Machine	60
3.6 Variable Access	64
3.6.1 Model Description	64
3.6.2 Variable Access Objects	64
3.6.3 Variable Access Services	73
3.7 Event Management	82
3.7.1 Model Description	82
3.7.2 The Event Object	83
3.7.3 Event Management Services	84
3.7.4 Event Notification State Machine	87
3.7.5 Examples	87

		Page
4	Interface between FMS and FAS	88
4.1	Overview of Services	88
4.2	Mapping of FMS Services on to FAS Services	89
5	Operating Behavior of FMS	89
5.1	Start of FMS	89
5.2	Conditions of Readiness for Operation	89
5.3	FMS Readiness for Operation	89
6	Structured Parameters ErrorType and Type Description	89
6.1	Parameter Error Type	89
6.1.2	Meaning of Error Class and Error Code	89
6.1.3	Meaning of the remaining Parameters	91
6.2	Parameter Type Description	91
7	Tables	92
7.1	List of Object Codes	92
7.2	List of Standard Data Types	92
7.3	List of Object Attributes and Service Parameters	93
7.4	List of Services	94
8	Conformance	95
8.1	Implementation and System (PICS Part 1)	95
8.2	Supported Services (PICS Part 2)	95
8.3	FMS Parameters and Options (PICS Part 3)	96
8.4	Local Implementation Values (PICS Part 4)	96
9	Coding	96
9.1	General	96
9.2	Coding Rules	96
9.3	Structure of the Identification Information	97
9.3.1	Coding of Application Data	98
9.3.2	Coding of Structure Information	104
10	Syntax Description	105
10.1	The FMS PDU	105
10.1.1	Description of the fixed PDU Part	105
10.1.2	ConfirmedServiceRequest	107
10.1.3	ConfirmedServiceResponse	108
10.1.4	ServiceError	109
10.1.5	VFD Support	112
10.1.6	OD Management	113
10.1.7	Context Management	114
10.1.8	Domain Management	115
10.1.9	Program Invocation Management	118
10.1.10	Variable Access	120
10.1.11	Event Management	123
10.1.12	List of packed fields	124
10.1.13	Detailed Coding Examples (Informative)	125
Figure 1 — General DD 238 Architecture		1
Figure 2 — VFD Model		2
Figure 3 — Abstract Model of an Automation System (VFD)		2
Figure 4 — VFD Support Services		2
Figure 5 — Source OD/Remote OD		7

	Page
Figure 6 — State Machine	17
Figure 7 — Example of a PutOD Sequence	18
Figure 8 — Transaction Object State Machine	22
Figure 9 — Context test of two FMS-Features-Supported with different bitstring length	27
Figure 10 — Connection Establishment State Machine	28
Figure 11 — Domain GenericDownload/Download StateMachine (Server)	48
Figure 12 — Domain Upload State Machine (Server)	50
Figure 13 — Example of an GenericDownload Sequence	51
Figure 14 — Example of a Download Sequence	51
Figure 15 — Example of an Upload Sequence	52
Figure 16 — Overview of the Program Invocation	53
Figure 17 — State Diagram	61
Figure 18 — Overview of Simple Variable	65
Figure 19 — Overview of Array	67
Figure 20 — Overview of Record	68
Figure 21 — Overview of Variable List	70
Figure 22 — Overview of Data Type Structure Description	73
Figure 23 — Overview of Event	82
Figure 24 — Event State Machine	87
Figure 25 — Insertion of Identification Information in the FMS PDU	96
Figure 26 — Coding of the ID Info with and without Extension	98
Figure 27 — Coding of ID Info for a SEQUENCE	104
Figure 28 — Structure of an Object Description in the OD	124
Figure 29 — Structure of the OD Object Description	124
Figure 30 — Structure of a Domain Entry in the S-OD	124
Figure 31 — Object Description of Program Invocation in the DP-OD	124
Figure 32 — Object Description of Simple Variable in the S-OD	124
Figure 33 — Object Description of Array in the S-OD	125
Figure 34 — Object Description of Record in the S-OD	125
Figure 35 — Object Description of Variable List in the DV-OD	125
Figure 36 — Object Description of Data Type in the ST-OD	125
Figure 37 — Object Description of Data Type Structure Description in the ST-OD	125
Figure 38 — Object Description of Event in the S-OD	125
Table 1 — Status	4
Table 2 — UnsolicitedStatus	5
Table 3 — Identify	5
Table 4 — Structure of the Object Dictionary	7
Table 5 — Structure of the Static List of Types	8
Table 6 — Structure of the Static Object Dictionary	8
Table 7 — Structure of the Dynamic List of Variable Lists	9
Table 8 — Structure of the Dynamic List of Program Invocations	9
Table 9 — Empty Object Dictionary	11
Table 10 — GetOD	13
Table 11 — InitiatePutOD	14

	Page
Table 12 — PutOD	15
Table 13 — Empty Object Description (Example)	15
Table 14 — TerminatePutOD	16
Table 15 — Attribute FMS Features Supported	20
Table 16 — Initiate	22
Table 17 — Abort	24
Table 18 — Reject	25
Table 19 — Compatibility of the local FMS Context to the remote FMS Context	26
Table 20 — Access Groups for a Domain	38
Table 21 — Access Rights for Domains	39
Table 22 — GenericInitiateDownloadSequence	40
Table 23 — GenericDownloadSegment	40
Table 24 — GenericTerminateDownloadSequence	41
Table 25 — InitiateDownloadSequence	42
Table 26 — DownloadSegment	43
Table 27 — TerminateDownloadSequence	43
Table 28 — RequestDomainDownload	44
Table 29 — InitiateUploadSequence	45
Table 30 — UploadSegment	45
Table 31 — TerminateUploadSequence	46
Table 32 — RequestDomainUpload	47
Table 33 — Access Groups for Program Invocation	54
Table 34 — Access Rights for PI	54
Table 35 — Create Program Invocation	55
Table 36 — Delete Program Invocation	56
Table 37 — Start	57
Table 38 — Stop	58
Table 39 — Resume	58
Table 40 — Reset	59
Table 41 — Kill	60
Table 42 — Variable Access Services and Objects	64
Table 43 — Access Groups for Simple Variable	66
Table 44 — Access Rights for Simple Variable	66
Table 45 — Access Groups for Record	69
Table 46 — Access Rights for Record	69
Table 47 — Access Groups for Variable List	71
Table 48 — Access Rights for Variable List	72
Table 49 — Read	74
Table 50 — Write	74
Table 51 — PhysRead	75
Table 52 — PhysWrite	76
Table 53 — InformationReport	76
Table 54 — DefineVariableList	77
Table 55 — DeleteVariableList	78
Table 56 — ReadWithType	79
Table 57 — WriteWithType	80
Table 58 — InformationReportWithType	81

	Page
Table 59 — Access Groups for Event	83
Table 60 — Access Rights for Events	84
Table 61 — EventNotification	84
Table 62 — AcknowledgeEventNotification	85
Table 63 — AlterEventConditionMonitoring	85
Table 64 — EventNotificationWithType	86
Table 65 — Example Event Services	88
Table 66 — Error Type	89
Table 67 — Type Description	91
Table 68 — Object Codes	92
Table 69 — Standard Data Types	92
Table 70 — Object Attributes and Service Parameters	93
Table 71 — Meaning of Services	94
Table 72 — Implementation and System	95
Table 73 — Supported Services	95
Table 74 — FMS Parameters and Options	96
Table 75 — Local Implementation Values	96
Table 76 — Representation of the Value true	98
Table 77 — Representation of the Value false	99
Table 78 — Coding of Data of Data Type Integer16	99
Table 79 — Coding of Data of Data Type Unsigned16	99
Table 80 — Coding of Data of Data Type Floating Point	100
Table 81 — Coding of Data of the Data Type Visible String	100
Table 82 — Coding of Data of the Data Type Octet String	101
Table 83 — Coding of Data of the Data Type Date	101
Table 84 — Coding of Data of the Data Type Time Of Day	102
Table 85 — Coding of Data of The Data Type Time Difference	103
Table 86 — Coding of Data of the Data Type Bit String	103
Table 87 — Coding of Data of the Data Type Time Value	104
Table 88 — Field with Fixed Format	106

Foreword

This Part of DD 238 has been prepared by Technical Committee, AMT/7. It supersedes DD 236-5:1996, which is withdrawn.

Portions of this document are owned by Fieldbus Foundation and are being used with their expressed written permission.

This publication contains the following parts:

- *Part 1: Guide to system architecture (Equivalent to FF-800 Revision 1.0¹⁾);*
- *Part 2: Specification for 31.25 kbit/s — Physical layer profile (Equivalent to FF-816 Revision 1.0¹⁾);*
- *Part 3: Specification for Data link layer services subset (Equivalent to FF-821 Revision 1.0¹⁾);*
- *Part 4: Specification for Data link protocol (Equivalent to FF-822 Revision 1.0¹⁾);*
- *Part 5: Specification for Fieldbus access sublayer (Services and protocol) (Equivalent to FF-875 Revision 1.0¹⁾);*
- *Part 6: Fieldbus Message Specification (Equivalent to FF-870 Revision 1.0¹⁾);*
- *Part 7: Fieldbus management;*
- *Section 7.1: Specification for System management (Equivalent to FF-880 Revision 1.0¹⁾);*
- *Section 7.2: Specification for Network management (Equivalent to FF-801 Revision 1.0¹⁾).*

This Draft for Development is not to be regarded as a British Standard.

It is being issued in the Draft for Development series of publications and is of a provisional nature because the subject matter is still under development requiring wider exposure. It should be applied on this provisional basis, so that information and experience of its practical application may be obtained.

A review of this Draft for Development will be carried out not later than 2 years after its publication. Notification of the start of the review period, with a request for the submission of comments from users of this Draft for Development, will be made in an announcement in the appropriate issue of *BSI Update*.

According to the replies received, the responsible BSI Committee will judge whether the Draft for Development can be converted into a British Standard or what other action should be taken.

Observations which it is felt should receive attention before the official call for comments will be welcomed.

These should be sent to the Secretary of BSI Technical Committee AMT/7 at BSI, 389 Chiswick High Road, London, W4 4AL, giving the document reference and clause number and proposing, where possible, an appropriate revision of the text.

¹⁾ Available from the Fieldbus Foundation, 9390 Research Boulevard, Suite II-250, Austin, Texas 78759, USA.

Disclaimer of warranties

This document is provided on an “as is” basis and may be subject to future additions, modifications, or corrections depending on the results of field trial testing. BSI and the Fieldbus Foundation hereby disclaims all warranties of any kind, express or implied, including any warranty of merchantability or fitness for a particular purpose, for this document. In no event will they be responsible for any loss or damage arising out of or resulting from any defect, error or omission in this document or from anyone’s use of or reliance on this document.

Summary of pages

This document comprises a front cover, an inside front cover, pages i to viii, pages 1 to 127 and a back cover.

This standard has been updated (see copyright date) and may have had amendments incorporated. This will be indicated in the amendment table on the inside front cover.

1 Scope

The Fieldbus Message Specification (FMS) is based on a series of object oriented models that provide distributed applications with the capabilities and services necessary for their implementation in an open, interoperable environment. This specification should be read in conjunction with Fieldbus Access Sublayer, Network Management and System Management Specifications.

2 Architecture

The following figure shows the relationship between the FMS and the other communication and application components of the architecture.

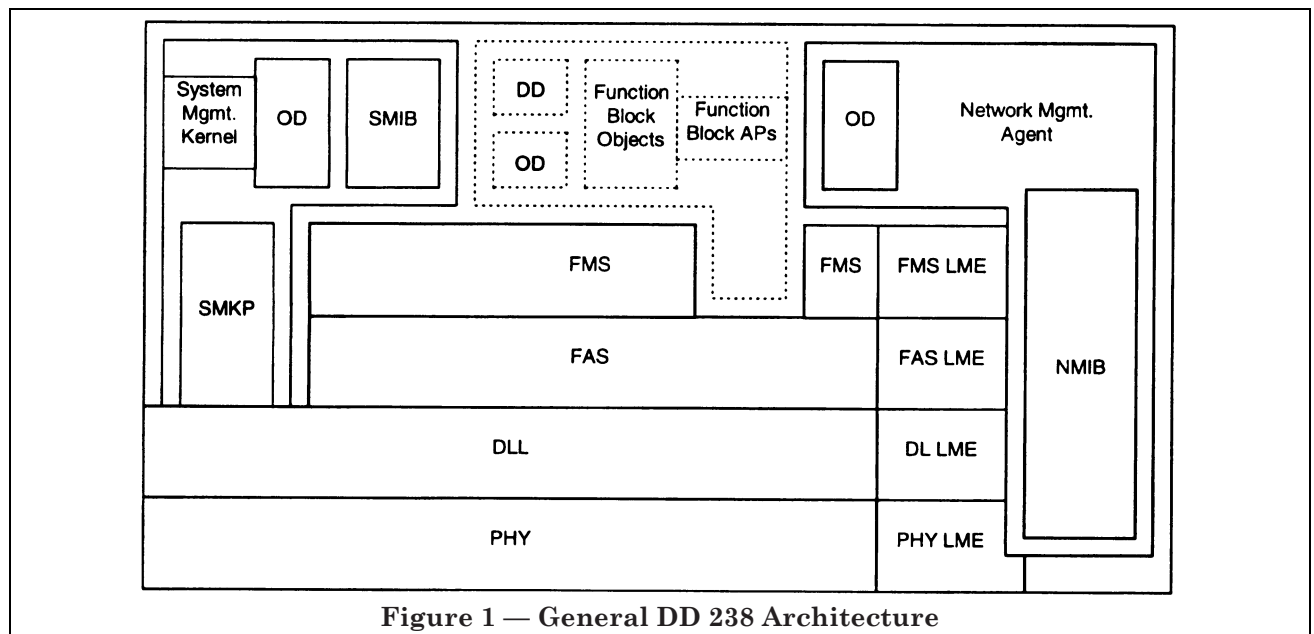


Figure 1 — General DD 238 Architecture

3 FMS Models and Services

This part presents the various models and their associated services. The models are:

- VFD Support
- Object Dictionary (OD) Management
- Context Management
- Domain Management
- Program Invocation Management
- Variable Access
- Event Management

All FMS services with their request and response parameters will be listed in figures. In these figures the parameter to reference VCR (Static VCR ID and Dynamic VCR ID) and Invoke-ID are missing. These parameters are service-independent in common, so they cannot be listed in a service description. But when a service request is transmitted at the interface to FMS, these parameters must be provided.

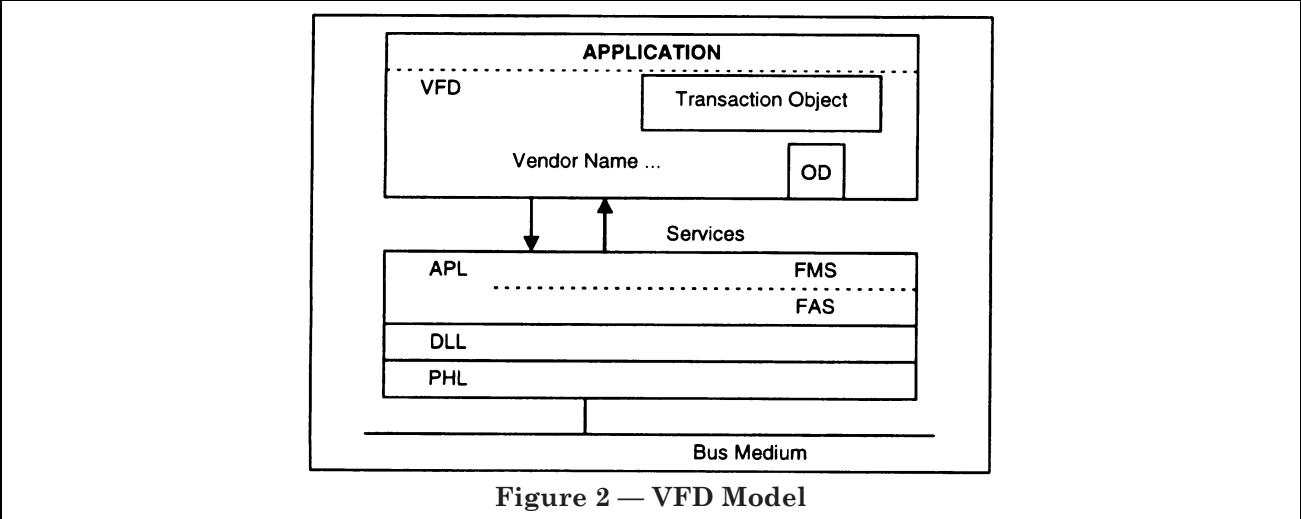
3.1 VFD Support

3.1.1 Model Description

The Virtual Field Device (VFD) is an abstract model for the description of the data and the behavior of an automation system as seen by a communication partner.

The basis of the VFD Model is the VFD Object. The VFD Object contains all objects and object descriptions which may be used by a communication user via services.

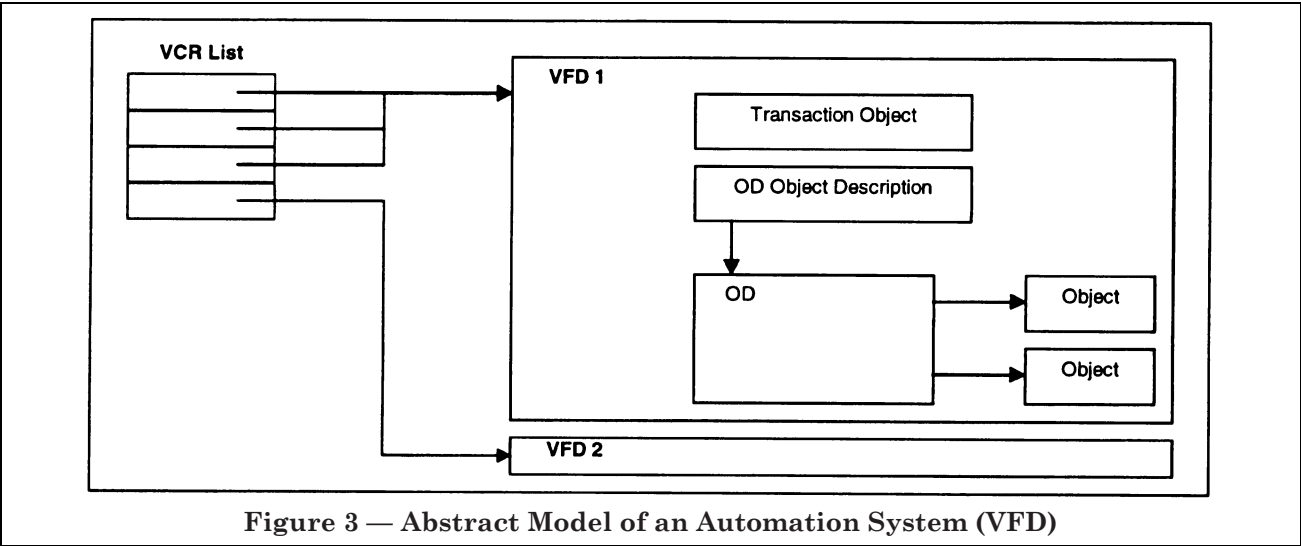
The object descriptions reside in the Object Dictionary (OD). For each VFD there is exactly one Object Dictionary.



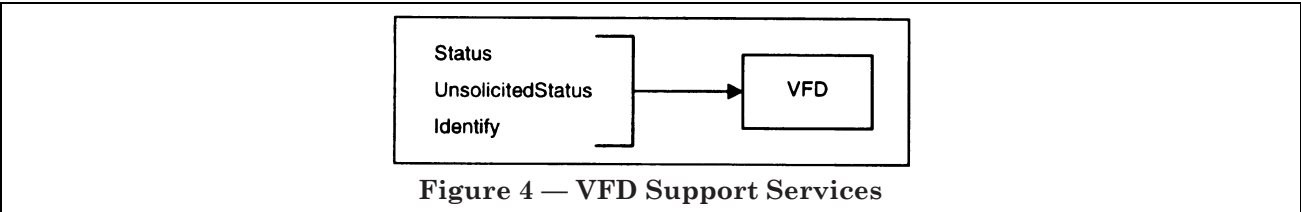
The FMS services define no concrete interface for an implementation. They describe in an abstract form which functions may be used.

The application is not a subject of this standard. It shall only be indicated, how the abstractly described services may be made available to the application.

The addressing of the VFD objects is defined implicitly by VCRs which are specified in VCR List as described in Network Management. Several VFD Objects may be present in a device.



The following VFD Support Services are defined:



Specific state information about the device may be given to the application with the Status/UnsolicitedStatus services. The Identify Service is used to identify the appropriate VFD on this VCR.

3.1.2 The VFD Object

The VFD Object is implicitly defined. The object description of the VFD may not be remotely read. The VFD Object is identified by the following attributes:

- VendorName
- ModelName
- Revision
- ProfileNumber

If there are several VFDs, it is clear from the entry in the VCR List, which VFD shall be addressed.

3.1.2.1 Attributes

Class:	VFD
1. (m)	<i>Key Attribute:</i> implicit through the VCR
2. (m)	<i>Attribute:</i> VendorName
3. (m)	<i>Attribute:</i> ModelName
4. (m)	<i>Attribute:</i> Revision
5. (m)	<i>Attribute:</i> ProfileNumber
6. (m)	<i>Attribute:</i> LogicalStatus
7. (m)	<i>Attribute:</i> PhysicalStatus
8. (m)	<i>Attribute:</i> List of VFD Specific Objects

Implicit through the VCR

The VCR which is entered in the VCR List points to the assigned VFD Object.

VendorName

This attribute, of type visible string, identifies the vendor of the device.

ModelName

In this visible string the model name of the device is entered by the vendor.

Revision

A visible string, which describes the revision level of the device. The value of this string is defined by the vendor.

ProfileNumber

The profile identification is entered in this attribute by the vendor in an octet string with a fixed length of 2 octets. If the device does not correspond to a profile, then the value "0" shall be entered in both octets.

Logical Status

This attribute contains information about the state of the communication capabilities of the device.

- 0 <=> ready for communication
- 2 <=> limited number of services
- 4 <=> OD-LOADING-NON-INTERACTING
- 5 <=> OD-LOADING-INTERACTING

3.1.2.1.1.1 Ready for communication

All services may be used normally.

3.1.2.1.1.2 Limited number of services

Limited number of services may be supported for some cases. A profile specifies which services are available.

3.1.2.1.1.3 OD-LOADING-NON-INTERACTING

If the Object Dictionary is in the state OD-LOADING-NON-INTERACTING, it is not allowed to execute the service InitiatePutOD.

3.1.2.1.1.4 OD-LOADING-INTERACTING

If the Object Dictionary is in the state OD-LOADING-INTERACTING, then all connections, except for the connection over which the InitiatePutOD Service was received, are locked and the establishment of a further connection shall be rejected. On this connection only the following services are allowed:

- Initiate
- Abort
- Reject
- Status
- Identify
- PhysRead
- PhysWrite
- GetOD
- InitiatePutOD
- PutOD
- TerminatePutOD

Physical Status

This attribute gives a coarse summary of the state of the real device.

- 0 <=> operational
- 1 <=> partially operational
- 2 <=> not operational
- 3 <=> needs maintenance

List of VFD Specific Objects

Contains all VFD Specific Objects.

3.1.3 VFD Support Services**3.1.3.1 Status**

The device/user state is read with the Status service.

Table 1 — Status

Parameter Name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
Result(+)			S	S(=)
LogicalStatus			M	M(=)
PhysicalStatus			M	M(=)
LocalDetail			U	U(=)
Result(-)			S	S(=)
ErrorType			M	M(=)

Argument

The argument does not contain service specific parameters.

Result(+)

The parameter Result(+) indicates that the service was successfully executed.

LogicalStatus

This parameter contains information about the state of the communication capabilities of the device.

PhysicalStatus

This parameter gives a coarse summary of the state of the real device.

LocalDetail

This parameter gives the local state of the application and the device. The meaning of the individual bits shall be defined by profiles.

Result(–)

The parameter Result(–) indicates that the service was not executed successfully.

Error-Type

This parameter contains information about the reason for the unsuccessful execution of the service.

3.1.3.2 UnsolicitedStatus

The UnsolicitedStatus service is used by the user for a spontaneous transmission of the device/user state.

Table 2 — UnsolicitedStatus

Parameter Name	.req	.ind
Argument	M	M(=)
LogicalStatus	M	M(=)
PhysicalStatus	M	M(=)
LocalDetail	U	U(=)

Argument

The Argument contains the service specific parameters of the service call.

LogicalStatus

This parameter contains information about the state of the communication capabilities of the device.

PhysicalStatus

This parameter gives a coarse summary of the state of the real device.

LocalDetail

This parameter indicates the local state of the application and the devices. The meaning of the individual bits shall be defined by profiles.

3.1.3.3 Identify

Information to identify a VFD is read with this service.

NOTE The attribute ProfileNumber of the VFD is transmitted with the Initiate service

Table 3 — Identify

Parameter Name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
Result(+)			S	S(=)
VendorName			M	M(=)
ModelName			M	M(=)
Revision			M	M(=)
Result(–)			S	S(=)
ErrorType			M	M(=)

Argument

The argument contains no service specific parameters.

Result(+)

The parameter Result(+) indicates that the parameters could be read.

VendorName

This parameter contains the vendor name in the VFD attribute.

ModelName

This parameter gives the model name.

Revision

This parameter contains the Revision attribute of the VFD.

Result(-)

The parameter Result(-) indicates that the service could not be executed.

ErrorType

This parameter contains information about the reason the service could not be executed.

3.2 Object Dictionary (OD) Management**3.2.1 Model Description**

The Object Dictionary (OD) contains the Object Descriptions of the following Communication Objects:

- DataType — DataTypeStructureDescription
- Domain — ProgramInvocation
- SimpleVariable — Array
- Record — VariableList
- Event

An object description “Object Dictionary” (OD object description) is uniquely assigned to the Object Dictionary. The OD object description contains information about the structure of the Object Dictionary.

The object descriptions are marked with an unique index of type Unsigned 16.

A name of type visible string may also be assigned to the following objects:

- Domain — ProgramInvocation
- SimpleVariable — Array
- Record — VariableList
- Event

The index or the name serves as a key to the object and the object description.

The services address on principle the object of the server, because the real object exists there.

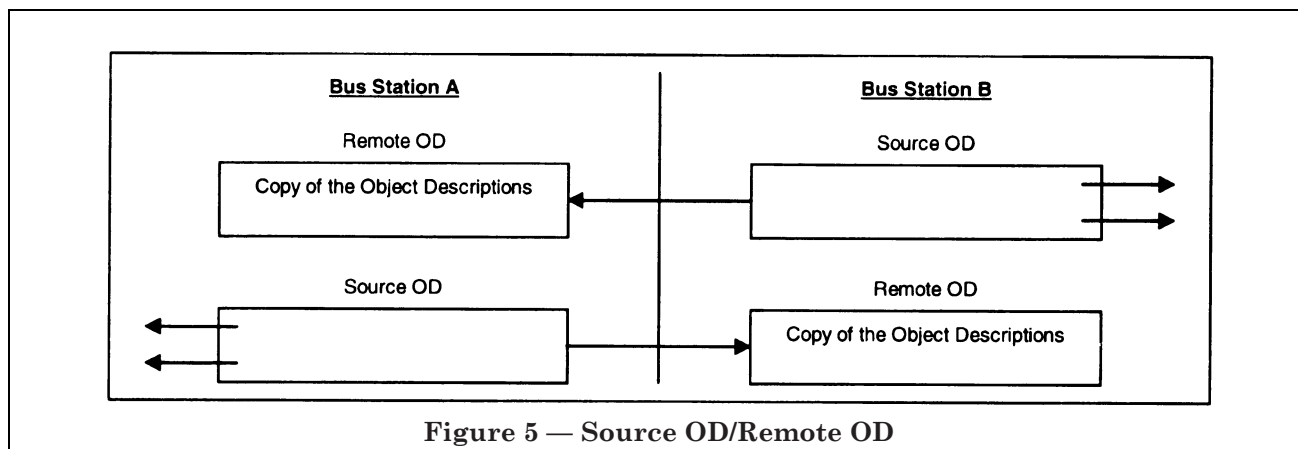
Before a remote communication partner can access an object, he shall know its description. The logical address (Index) or the name may be used, when objects are accessed remotely.

The object descriptions may be specified during system configuration, but they may also be transferred between stations at any later time.

The definition of the object description takes place at that station, in which the object really exists (Source OD). The length of an object description may be different for the individual objects. The communication partners maintain a complete or a partial copy of the Remote Object Description (Remote OD).

Thus every station may possess a Source OD for locally existing communication objects and one or more Remote ODs. The Remote ODs are connection specific, i.e. for each connection another Remote OD may be valid.

The OD model provides corresponding services for reading and writing of Source ODs. If a Source OD is modified by a remote communication partner, then this shall be notified to all other communication partners which use the same data base (Source OD). For this purpose, the user releases the connections to all other communication partners. After the OD has been modified, the connections may be established again. During the connection establishment, the new version number (Version OD) is given to each communication partner to notify that the corresponding Source OD has been modified. The addition of object descriptions is permissible without connection release.



The structure of the OD described as follows does not prescribe a concrete implementation of the OD. It only defines the structure of the object descriptions which are transmitted via the bus. Even the existence of an OD is not necessary, if the needed information can be obtained in an other way (e.g. by an algorithm). Detailed attributes and transfer syntaxes are specified in the appropriate models.

3.2.2 Structure of an OD

The Object Dictionary (OD) is divided into several parts:

- The Object description of the “Object Dictionary” (OD object description) contains structural information about the OD.
- Static Type Dictionary (ST-OD) contains Data Types and Data Type Structure Description.
- Static Object Dictionary (S-OD) contains object descriptions of Simple Variables, Arrays, Records, Domains and Events.
- Dynamic List of Variable Lists (DV-OD) contains object descriptions of variable lists.
- Dynamic List of Program Invocations (DP-OD) contains object descriptions of Program Invocations.

Table 4 — Structure of the Object Dictionary

Index	Object Dictionary (OD)
0	OD object description (OD-ODES)
1 ... i	Static List of Types (ST-OD)
k ... n	Static Object Dictionary (S-OD)
p ... t	Dynamic List of Variable Lists (DV-OD)
v ... z	Dynamic List of Program Invocations (DP-OD)

3.2.2.1 Static List of Types (ST-OD)

The Static List of Types (ST-OD) contains the object descriptions of Data Types and Data Type Structure Descriptions for the Variable Access objects of the S-OD.

Exactly one object description is assigned to every index. This assignment is static and may not be deleted. The ST-OD may be loaded by the remote partner.

The Static List of Types always begins with the index 1 in the Object Dictionary. The remaining object descriptions shall follow with increasing index. The number of assignments “index $\Leftrightarrow$ object description” is entered as length of the ST-OD in the Object Dictionary.

The ST-OD is divided in two segments. The first segment contains the object descriptions of standardized Data Types (Standard Data Types) and begins with index 1. The second segment follows immediately and contains object descriptions of Data Types and Data Type Structure Descriptions, which can be defined individually. This segment may be further specified by user groups with the help of profiles.

Table 5 — Structure of the Static List of Types

Index	Static List of Types (ST-OD)
1 ... c	contain objects: Data Types standardized by DD 238
d ... i	contains objects: Data Types and Data Type Structure Descriptions

3.2.2.2 Static Object Dictionary (S-OD)

The Static Object Dictionary (S-OD) contains the object descriptions of Simple Variables, Arrays, Records, Domains and Events.

An object description is assigned to each index. The first index, which is associated to an object description, is entered in the object description of the Object Dictionary. The remaining object descriptions shall follow with increasing index. The number of assignments “index $\Leftrightarrow$ object description” is entered as length of the S-OD in the object description of Object Dictionary.

Besides the assignment “index $\Leftrightarrow$ object description”, an additional assignment “name $\Leftrightarrow$ object description” may exist. The length of the name may be from 0 to 32 octets. It is entered in the Name Length Field of the OD object description. If this field contains a “0”, no names are present. The objects which are defined in the S-OD may be provided with access rights. Whether Access Rights are possible is entered in the field AccessProtectionSupported of the OD object description.

An object description in the S-OD contains an extension, which may contain further object specific definitions.

Table 6 — Structure of the Static Object Dictionary

Index	Static Object Dictionary (S-OD)
k ... n	contains objects: Simple Variables, Arrays, Records, Domains and Events

3.2.2.3 Dynamic List of Variable Lists (DV-OD)

The Dynamic List of Variable Lists (DV-OD) contains the object description of the Variable Lists.

An object “Variable List” and its object description in the DV-OD is dynamically created with the service DefineVariableList and may be deleted with the service DeleteVariableList. Upon creation of the object access rights may be assigned to it. Moreover it is possible, to create these objects by loading the whole OD.

The first index, which has the object description of a Variable List assigned to it, and the number of available indices are entered in the OD object description.

Fundamental information is entered in the object description about the Variable List, for example access rights, the number of Variable Access objects and the logical addresses (indices) of Variable Access objects in the S-OD.

Besides the assignment “index $\Leftrightarrow$ object description”, an additional assignment “name $\Leftrightarrow$ object description” may exist.

Table 7 — Structure of the Dynamic List of Variable Lists

Index	Dynamic List of Variable Lists (DV-OD)
p ... t	contains objects: Variable List

3.2.2.4 Dynamic List of Program Invocations (DP-OD)

The Dynamic List of Program Invocations (DP-OD) contains the object descriptions of the objects Program Invocation.

An object Program Invocation and its object description in the DP-OD is dynamically created with the service Create Program Invocation and deleted with the service Delete Program Invocation. Upon creation of the object access rights may be assigned to it. Moreover it is possible to create these objects by loading the whole OD.

The first index, which has the object description of a Program Invocation assigned to it, and the number of available indices are entered in the OD object description.

The object description of a Program Invocation contains information such as access rights and the number of Domain objects and their logical addresses (indices).

Besides the assignment “index $\Leftrightarrow$ object description”, an additional assignment “name $\Leftrightarrow$ object description” may exist.

The DP-OD may contain a segment with pre-configured Program Invocations.

Table 8 — Structure of the Dynamic List of Program Invocations

Index	Dynamic List of Program Invocations (PD-OD)
v ... w	Predefined Program Invocations
x ... z	Dynamic Program Invocations

3.2.3 Object Description in the OD

The object descriptions are entered in the OD. Every object description contains an index, an object code, further object attributes, system specific references to the real object and, if need be, a name and an extension.

The object is addressed by the index and identified by the object description. The object code is the identifier of the object and indicates which class this object belongs to. The other object attributes are object specific.

The local address serves the internal addressing of the object. Besides the index, optionally a name for the addressing of the object may also be used. The extension contains a length information for the extension and optionally further object attributes. The value “0” for the length means: no further object attributes.

3.2.4 OD Object Description

The structure of the Object Dictionary is described by the object description “Object Dictionary” (OD object description). This has the index “0” in the Object Dictionary. By reading the OD object description the structure description and the version number of the OD are made available.

3.2.4.1 Attributes

Class:	OD Object Description
1. (m)	<i>Key Attribute:</i> Numeric Identifier
2. (m)	<i>Attribute:</i> ROM/RAM-Flag
3. (m)	<i>Attribute:</i> NameLength
4. (m)	<i>Attribute:</i> AccessProtectionSupported
5. (m)	<i>Attribute:</i> VersionOD
6. (m)	<i>Attribute:</i> LocalAddress-OD-ODES

7.	(m)	<i>Attribute:</i>	ST-OD-Length
8.	(m)	<i>Attribute:</i>	LocalAddress-ST-OD
9.	(m)	<i>Attribute:</i>	FirstIndex-S-OD
10.	(m)	<i>Attribute:</i>	S-OD-Length
11.	(m)	<i>Attribute:</i>	LocalAddress-S-OD
12.	(m)	<i>Attribute:</i>	FirstIndex-DV-OD
13.	(m)	<i>Attribute:</i>	DV-OD-Length
14.	(m)	<i>Attribute:</i>	LocalAddress-DV-OD
15.	(m)	<i>Attribute:</i>	FirstIndex-DP-OD
16.	(m)	<i>Attribute:</i>	DP-OD-Length
17.	(m)	<i>Attribute:</i>	LocalAddress-DP-OD

Numeric Identifier

Identifies an instance of this object class, here always = 0.

ROM/RAM-Flag

This attribute of type Boolean describes whether modifications in the OD are allowed.

false <=> no modifications in the OD are permitted

true <=> modifications in the OD are permitted

NameLength

This field gives the length of the name and may take only the values “0..32”:

0 <=> no names are used

AccessProtectionSupported

This attribute states, whether access rights for password, access groups and all communication partners are supported (true), or whether the access to all objects is permitted for every communication partner (false).

VersionOD

Gives the version of the OD.

LocalAddress-OD-ODES

This attribute is a system specific reference to the real OD object description and serves the internal addressing. If no representation of this kind is used, this attribute is set to the value FFFFFFFF hex.

ST-OD-Length

This attribute gives the maximal number of available entries in the Static List of Types.

NOTE The first available index of the Static List of Types is always = 1, therefore the highest available index in the Static List of Types is equal to its length.

LocalAddress-ST-OD

This attribute is a system specific reference to the real ST-OD and serves the internal addressing. If no representation of this kind takes place, this attribute is set to the value FFFFFFFF hex.

FirstIndex-S-OD

The first available index in the Static Object Dictionary.

S-OD-Length

This attribute gives the maximum number of available entries in the Static Object Dictionary.

NOTE The highest available index is $S\text{-}OD\text{-}Length = FirstIndex\text{-}S\text{-}OD + S\text{-}OD\text{-}Length - 1$ (done)

LocalAddress-S-OD

This attribute is a system specific reference to the real S-OD and serves the internal addressing. If no representation of this kind takes place, this attribute is set to the value FFFFFFFF hex.

FirstIndex-DV-OD

First available index in the Dynamic List of Variable Lists.

0 <=> no Dynamic List of Variable Lists.

DV-OD-Length

This attribute gives the maximum number of available entries in the Dynamic List of Variable Lists.

NOTE The highest available index is $DV-OD-Length = FirstIndex-DV-OD + OD-OD-Length - 1$

LocalAddress-DV-OD

This attribute is a system specific reference to the real DV-OD and serves the internal addressing. If no representation of this kind takes place, this attribute is set to the value FFFFFFFF hex.

FirstIndex-DP-OD

First available index in the Dynamic List of Program Invocations.

0 <=> no Dynamic List of Program Invocations.

DP-OD-Length

This attribute gives the maximum number of available entries in the Dynamic List of Program Invocations.

NOTE The highest available index is $DP-OD-Length = FirstIndex-DP-OD + DP-OD-Length - 1$

LocalAddress-DP-OD

This attribute is a system specific reference to the real DP-OD and serves the internal addressing. If no representation of this kind takes place, this attribute is set to the value FFFFFFFF hex.

3.2.4.2 Empty Object Dictionary

The Object Dictionary is loadable. Therefore if an Object Dictionary is not yet loaded, at least an empty Object Dictionary shall exist.

An empty Object Dictionary is defined by pre-setting the following attributes of the OD object description:

Table 9 — Empty Object Dictionary

Attribute	Value
ROM/RAM-Flag	True
NameLength	0
AccessProtectionSupported	False
VersionOD	0
ST-OD-Length	0
FirstIndex-S-OD	0
S-OD-Length	0
FirstIndex-DV-OD	0
DV-OD-Length	0
FirstIndex-DP-OD	0
DP-OD-Length	0

3.2.5 The OD Object**3.2.5.1 Attributes**

Class: OD

1. (m) *Key Attribute:* implicit by VFD
2. (m) *Attribute:* OD object description
3. (m) *Attribute:* List of object description

implicit by VFD

The VCR, which is entered in the VCR List points to the assigned Object Dictionary.

OD object description

Contains the object description of the Object Dictionary.

List of object description

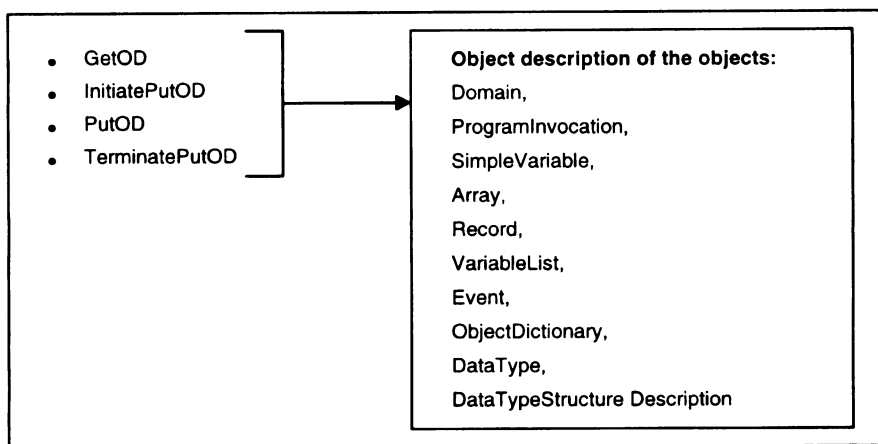
Contains object descriptions of the following objects:

— Domain — VariableList

- ProgramInvocation — DataType
- SimpleVariable — DataTypeStructureDescription
- Array — Event
- Record

3.2.6 OD Services

The OD services act on the object descriptions of the objects Domain, Program Invocation, Simple Variable, Array, Record, Variable List, Event, Object Dictionary, Data Type and Data Type Structure Description. The following OD services are defined:



With the GetOD service one or several object descriptions are read. With the PutOD service one or several object descriptions are written. The number of object descriptions, which are transferable with one PutOD or GetOD Service, depends on their length and the maximum available PDU size.

Write access to the OD shall be initiated with an InitiatePutOD service and terminated with a TerminatePutOD service.

3.2.6.1 GetOD

With the GetOD service one or several object descriptions are read. The service distinguishes between a short and a long form. The long form of the GetOD service is optional. To read a single object description, the index or the name shall be given.

To read several or all object descriptions, the index of the first required object description (StartIndex) shall be given. To read the whole OD, the GetOD service shall be used repeatedly.

Argument

AllAttributes

This parameter states whether the object description shall be transferred in the short form (false) or in the long form (true).

The following object attributes are not contained in the short form:

- Password — LocalAddress-OD-ODES
- AccessGroups — LocalAddress-ST-OD
- AccessRights — LocalAddress-S-OD
- LocalAddress — LocalAddress-DV-OD
- Name — LocalAddress-DP-OD
- Extension

AccessSpecification

This parameter states which object description is accessed.

Index

Index of the object description.

Table 10 — GetOD

Parameter Name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
AllAttributes	M	M(=)		
AccessSpecification	M	M(=)		
Index	S	S(=)		
VariableName	S	S(=)		
VariableListName	S	S(=)		
DomainName	S	S(=)		
PI-Name	S	S(=)		
EventName	S	S(=)		
StartIndex	S	S(=)		
Result(+)			S	S(=)
List of Object Description			M	M(=)
MoreFollows			M	M(=)
Result(-)			S	S(=)
ErrorType			M	M(=)

VariableName

Name of a variable.

VariableListName

Name of a Variable List.

PI-Name

Name of a Program Invocation.

DomainName

Name of a Domain.

EventName

Name of an Event.

StartIndex

Index, starting with which the object descriptions shall be read.

Result(+)

The Result(+) parameter indicates that the service was executed successfully.

List of object description

For logical access and for access by name one, otherwise several complete object descriptions. The number depends on the maximum possible PDU size and the length of the single object descriptions. If access rights are not supported (Access Protection = false), the attributes Password, Access Groups and Access Rights are meaningless.

MoreFollows

States whether further object descriptions exist after reading several object descriptions (StartIndex). For the access to single object descriptions this parameter is always = false.

Result(-)

The Result(-) parameter indicates that the service was not executed successfully.

ErrorType

This parameter contains information about the reason for the unsuccessful execution of the service.

3.2.6.2 PutOD Services

With the PutOD services one or several object descriptions are written. Before the beginning of a modification of the OD an user may take appropriate measures (e.g. to put a process into a safe state). The communication does not test the plausibility of the new Object Dictionary.

Writing in the OD is performed with the sequence InitiatePutOD service, one or several PutOD services and a TerminatePutOD service. For this it is distinguished between a loading which is free of interactions and a loading which is not free of interactions (OD-LOADING-NON-INTERACTING and OD-LOADING-INTERACTING). The loading which is not free of interactions differentiates between a new loading (complete) and a post loading (partial). If need be, new communication objects and their state machines are created with the TerminatePutOD.

Loading free of interactions (OD-LOADING-NON-INTERACTING):

The client communicates to the server with the parameter Consequence = 0, whether the intended modifications or extensions are free of interactions for the other communication partners. In this case the VCRs are continued. For example, modifications free of interactions are:

- Addition of new object descriptions in the static segment
- Deletion of private object descriptions for stations which are no longer in the communication system

The communication does not check whether the write to the OD is really free of interactions.

The loading free of interaction is initiated with the service InitiatePutOD with the parameter Consequence = 0. After that object descriptions may be loaded with PutOD. As long as the loading process is not terminated with TerminatePutOD, every further InitiatePutOD is rejected.

Loading not free of interactions (OD-LOADING-INTERACTING):

The loading not free of interactions is initiated with the service InitiatePutOD with the Parameter Consequence = 1 (post-loading) or Consequence = 2 (new-loading). In this case the user may abort all active VCRs, except the one over which the InitiatePutOD is received to this VFD. After that the object description may be loaded with PutOD. It is also the task of the client user to enter the new version number in the OD. The loading process is terminated with the service TerminatePutOD. After that all VCRs may be re-established. With the re-establishment of the VCRs the new version number of the OD is transferred. The communication partners may then recognize that the corresponding OD has been modified. Whereas for post-loading solely parts of the OD are loaded, the existing OD is deleted and subsequently loaded completely anew upon receipt of an InitiatePutOD with Consequence = 2.

3.2.6.3 InitiatePutOD

The InitiatePutOD opens the process of loading the OD free of interaction or not free of interaction.

Table 11 — InitiatePutOD

Parameter Name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
Consequence	M	M(=)		
Result(+)			S	S(=)
Result(-)			S	S(=)
Error Type			M	M(=)

Argument

The argument contains the service specific parameters of the service call.

Consequence

This parameter states whether the intended modifications are free of interaction for the other communication partners (Consequence = 0), not free of interaction with partial reload of the OD (Consequence = 1) or not free of interaction with completely new load of the OD (Consequence = 2).

- 0 <=> loading free of interaction
- 1 <=> reload, not free of interaction
- 2 <=> new load, not free of interaction

Result(+)

The Result(+) parameter indicates that the service was executed successfully.

Result(-)

The Result(-) parameter indicates that the service was not executed successfully.

ErrorType

This parameter contains information about the reason for the unsuccessful execution of the service.

3.2.6.4 PutOD

With the PutOD service one or several object descriptions are written into the OD.

Table 12 — PutOD

Parameter Name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
List of Object Description	M	M(=)		
Result(+)			S	S(=)
Result(-)			S	S(=)
Error Type			M	M(=)

Argument

The argument contains the service specific parameters of the service call.

List of object description

One or several object descriptions which shall be entered into the OD. If access rights are not supported (Access Protection = false), the attributes Password, Access Groups and Access Rights are meaningless.

An object description may also be deleted with the PutOD service. In this case, an empty object description with the following structure is transferred:

- Index of the object description to be deleted
- Object Code = Null Object.

Table 13 — Empty Object Description (Example)

Index	Object Code
246	Null Object

Result(+)

The Result(+) parameter indicates that the service was successfully executed.

Result(-)

The Result(-) parameter indicates that the service was not successfully executed. In this case, no object description is written into the OD.

ErrorType

This parameter contains information about the reason for the unsuccessful execution of the service.

3.2.6.5 TerminatePutOD

The TerminatePutOD service terminates the process of loading of the OD. The objects with their state machines are generated.

Table 14 — TerminatePutOD

Parameter Name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
Result(+)			S	S(=)
Result(-)			S	S(=)
Error Type			M	M(=)
Index			C	C(=)

Argument

The Argument contains no service specific parameters.

Result(+)

The Result(+) parameter indicates that the service was successfully executed.

Result(-)

The Result(-) parameter indicates that the service was not successfully executed. If an error occurs during the generation of the objects, the old state is restored.

ErrorType

This parameter contains information about the reason for the unsuccessful execution of the service.

Index

This parameter gives the index of the object, for which the generation was not successful.

3.2.7 State Machine**3.2.7.1 State Machine Description****OD-LOADING-NON-INTERACTING**

In this state the execution of the service InitiatePutOD is not permitted. In this state the service PutOD on the DV-OD or the DP-OD is not permitted.

OD-LOADING-INTERACTING

Except for the connection over which the InitiatePutOD service was received, all connections are locked and the establishment of a further connection shall be rejected. On this connection only the following services are allowed:

- | | |
|------------|------------------|
| — Initiate | — PhysWrite |
| — Abort | — GetOD |
| — Reject | — InitiatePutOD |
| — Status | — PutOD |
| — Identify | — TerminatePutOD |
| — PhysRead | |

OD-READY

In this state all connections and services may be used normally.

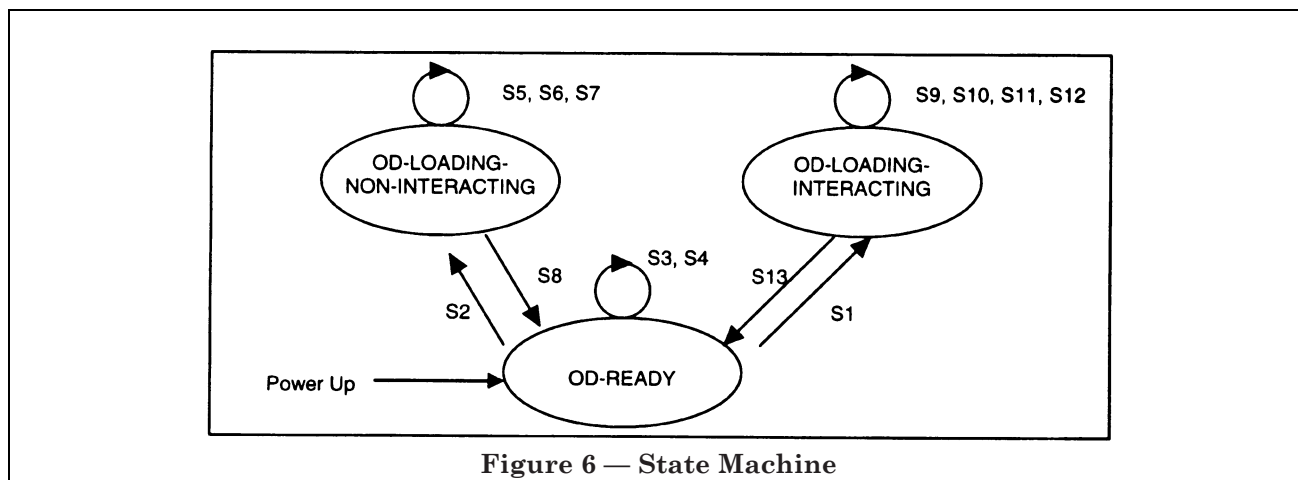
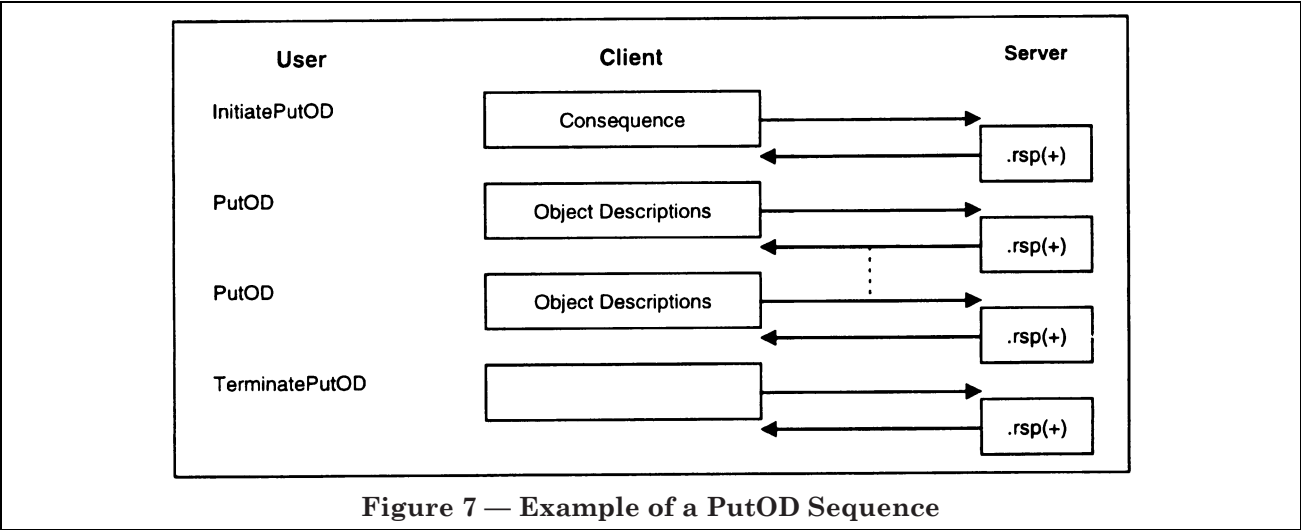


Figure 6 — State Machine

3.2.7.2 State Transitions

#	Current State	Events	Actions	Next State
S1	OD-READY	InitiatePutOD.ind && consequence <> loading-free-of-interaction	InitiatePutOD.rsp(+){ }	OD-LOADING-INTERACTING
S2	OD-READY	InitiatePutOD.ind && consequence = loading-free-of-interaction	InitiatePutOD.rsp(+){ }	OD-LOADING-NON-INTERACTING
S3	OD-READY	PutOD.ind	PutOD.rsp(-){ ErrorType = Object-state-conflict }	OD-READY
S4	OD-READY	TerminatePutOD.ind	TerminatePutOD.rsp(-){ ErrorType = Object-state-conflict }	OD-READY
S5	OD-LOADING-NON-INTERACTING	InitiatePutOD.ind	InitiatePutOD.rsp(-){ ErrorType = object-state-conflict }	OD-LOADING-NON-INTERACTING
S6	OD-LOADING-NON-INTERACTING	PutOD.ind	PutOD.rsp(+){ }	OD-LOADING-NON-INTERACTING
S7	OD-LOADING-NON-INTERACTING	TerminatePutOD.ind && OD is not usable	TerminatePutOD.rsp(-){ ErrorType = operational-problem }	OD-LOADING-NON-INTERACTING
S8	OD-LOADING-NON-INTERACTING	TerminatePutOD.ind && OD is okay	TerminatePutOD.rsp(+){ }	OD-READY
S9	OD-LOADING-INTERACTING	InitiatePutOD.ind && consequence <> loading-free-of-interaction	InitiatePutOD.rsp(+){ }	OD-LOADING-INTERACTING
S10	OD-LOADING-INTERACTING	InitiatePutOD.ind && consequence = loading-free-of-interaction	InitiatePutOD.rsp(-){ ErrorType = object-state-conflict }	OD-LOADING-INTERACTING
S11	OD-LOADING-INTERACTING	PutOD.ind	PutOD.rsp(+){ }	OD-LOADING-INTERACTING
S12	OD-LOADING-INTERACTING	TerminatePutOD.ind && OD is not usable	TerminatePutOD.rsp(-){ ErrorType = operational-problem }	OD-LOADING-INTERACTING
S13	OD-LOADING-INTERACTING	TerminatePutOD.ind && OD is okay	TerminatePutOD.rsp(+){ }	OD-READY

3.2.7.3 Example of a PutOD



3.3 Context Management

3.3.1 Model Description

The context contains all agreements concerning a VCR. The context management services shall be used to establish a connection, to release a connection and to reject improper services. The data necessary for this are stored in the OD object description and in the VCR List. Transaction objects are provided for each VCR. The transaction objects are used for the management of confirmed service primitives.

3.3.2 The VCR List

The Virtual Communication Relationship List (VCR List) contains the descriptions of all VCRs of a device irrespective of the time of use. A VCR consists of static part and dynamic part, each of which is a structure data. This section gives the definition to FMS specific part of VCR. Refer Network Management Specification for the definition of other part of VCR.

Class:		Static VCR Information	
1.	(m)	Key Attribute:	Static VCR ID
2.	(m)	Attribute:	Max FMS PDU Sending
3.	(m)	Attribute:	Max FMS PDU Receiving
4.	(m)	Attribute:	FMS Features Supported Length
5.	(m)	Attribute:	FMS Features Supported
6.	(m)	Attribute:	Max Outstanding Services Sending
7.	(m)	Attribute:	Max Outstanding Services Receiving
8.	(m)	Attribute:	FMS VFD ID
Class:		Dynamic VCR Information	
1.	(m)	Key Attribute:	Dynamic VCR ID
2.	(m)	Attribute:	Outstanding Services Counter Sending
3.	(m)	Attribute:	Outstanding Services Counter Receiving
4.	(m)	Attribute:	FMS State
5.	(m)	Attribute:	Actual Max Outstanding Services Sending
6.	(m)	Attribute:	Actual Max Outstanding Services Receiving

VCR Static Information

Static VCR ID

This parameter gives the index in the Object Dictionary to the static part of this VCR.

Max FMS PDU Sending

This attribute shall specify the maximum length of the FMS PDU to be sent that may be handled on this VCR. This attribute is not actually a part of VCR but is calculated from other parameters at any time.

Max FMS PDU Receiving

This attribute shall specify the maximum length of the FMS PDU to be received that may be handled on this VCR. This attribute is not actually a part of VCR but is calculated from other parameters at any time.

FMS Features Supported Length

This attribute defines the number of octets present in the FMS Features Supported field.

FMS Features Supported

This attribute shall identify the optional FMS services and the optional features supported on this VCR. This section specifies bit assignment of fixed part and the method for extension of services and features. The FMS Features Supported is of the length specified by FMS Features Supported Length. It consists of two concatenated bit strings of the same length. The first bit string identifies services and features as a client and the second bit string identifies services and features as a server. For each optional service there are two bits, one in the first bit string and another in the second bit string. For the optional feature there are two bits, one in the first bit string, specifying the optional feature is permissible. The extension of the attribute can be made by using unassigned bits or adding another octet at the end of each bit string. All bits within a byte, which are not specified, have the value 0.

Following services are mandatory for all VCRs to open and close them:

- Initiate
- Abort

For consistency, the server supports at least the following basic services:

- Identify
- Status
- Reject
- GetOD

Max Outstanding Services Sending (MaxSCC)

This attribute shall specify the maximum number of outstanding confirmed services capable in this station as the requester side of this VCR. This attribute is used on client-server and peer-to-peer VCR (i.e., QUB) and corresponds to the number of sending state machines.

Max Outstanding Services Receiving (MaxRCC)

This attribute shall specify the maximum number of outstanding confirmed services capable in this station as the responder side of this VCR. This attribute is used on client-server and peer-to-peer VCR (i.e., QUB) and corresponds to the number of receiving state machines.

FMS VFD ID

This pointer shall refer to the assigned VFD.

Dynamic VCR Information

Dynamic VCR ID

This parameter gives the index in the Object Dictionary to the dynamic part of this VCR.

Outstanding Services Counter Sending (OSCS)

This attribute shall specify the number of currently outstanding confirmed services on this VCR as a requester.

Outstanding Services Counter Receiving (OSCR)

This attribute shall specify the number of outstanding services that are currently being processed on this VCR as a responder.

Table 15 — Attribute FMS Features Supported

Service	Primitive of client functions	bit(n)	Primitive of server functions	bit(n)
GetOD (Long form)	.req, .cnf	0	.ind, .rsp	0
UnsolicitedStatus	.req	1	.ind	1
InitiatePutOD	.req, .cnf	2	.ind, .rsp	2
PutOD	.req, .cnf	"	.ind, .rsp	"
TerminatePutOD	.req, .cnf	"	.ind, .rsp	"
InitiateDownloadSequence	.req, .cnf	3	.ind, .rsp	3
DownloadSegment	.ind, .rsp	"	.req, .cnf	"
TerminateDownloadSequence	.ind, .rsp	"	.req, .cnf	"
InitiateUploadSequence	.req, .cnf	4	.ind, .rsp	4
UploadSegment	.req, .cnf	"	.ind, .rsp	"
TerminateUploadSequence	.req, .cnf	"	.ind, .rsp	"
RequestDomainDownload	.req, .cnf	5	.ind, .rsp	5
RequestDomainUpload	.req, .cnf	6	.ind, .rsp	6
CreateProgramInvocation	.req, .cnf	7	.ind, .rsp	7
DeleteProgramInvocation	.req, .cnf	"	.ind, .rsp	"
Start	.req, .cnf	8	.ind, .rsp	8
Stop	.req, .cnf	"	.ind, .rsp	
Resume	.req, .cnf		.ind, .rsp	
Reset	.req, .cnf		.ind, .rsp	"
Kill	.req, .cnf	9	.ind, .rsp	9
Read	.req, .cnf	10	.ind, .rsp	10
Write	.req, .cnf	11	.ind, .rsp	11
ReadWithType	.req, .cnf	12	.ind, .rsp	12
WriteWithType	.req, .cnf	13	.ind, .rsp	13
PhysRead	.req, .cnf	14	.ind, .rsp	14
PhysWrite	.req, .cnf	15	.ind, .rsp	15
InformationReport	.req	16	.ind	16
InformationReportWithType	.req	17	.ind	17
DefineVariableList	.req, .cnf	18	.ind, .rsp	18
DeleteVariableList	.req, .cnf	"	.ind, .rsp	"
EventNotification	.req	19	.ind	19
EventNotificationWithType	.req	20	.ind	20
AcknowledgeEventNotification	.req, .cnf	21	.ind, .rsp	21
AlterEventConditionMonitoring	.req, .cnf	22	.ind, .rsp	22
	Primitive of client functions	bit[n]	Primitive of server functions	bit[n]
Addressing with Name	.req	23	.ind	23
	Primitive of client functions	bit[n]	Primitive of server functions	bit[n]
Extensions				
GenericInitiateDownloadSequence	.req, .cnf	24	.ind, .rsp	24
GenericDownloadSegment	.req, .cnf	"	.ind, .rsp	"
GenericTerminateDownloadSequence	.req, .cnf	"	.ind, .rsp	"
Explanation: [n] : 0 to (23+8x); without extension ==> x = 0; with extension ==> x = 1,2, ...				

FMS State

This attribute shall specify the state of the VCR. A connection-oriented VCR may be in one of the following states:

- CONNECTION-NOT-ESTABLISHED
- CONNECTION-ESTABLISHING (CALLING)
- CONNECTION-ESTABLISHING (CALLED)
- CONNECTION-ESTABLISHED

Actual Max Outstanding Services Sending (ActualMaxSCC)

This attribute shall specify the maximum number of outstanding confirmed services allowed in this station as the requester side of this VCR. Its value shall be negotiated during initialization of this VCR.

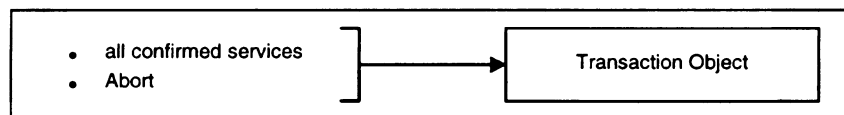
Actual Outstanding Services Receiving (ActualMaxRCC)

This attribute shall specify the maximum number of outstanding confirmed services allowed in this station as the responder side of this VCR. Its value shall be negotiated during initialization of this VCR.

3.3.3 The Transaction Object

A Transaction Object shall be created upon receipt of a confirmed service indication primitive. The Transaction Object shall be uniquely related to the corresponding service primitive. The Transaction Object shall be deleted after the corresponding response primitive has been sent. The Transaction Object shall be uniquely identified by the combination of Invoke ID and Static VCR ID.

The maximum number of Transaction Objects for each VCR of the server shall be specified by the Max Outstanding Services Receiving attribute in the VCR entry. The following services act on the Transaction Object:



3.3.3.1 Attributes

Class: Transaction Object

1. (m) *Key Attribute:* Invoke ID & VCR ID
2. (m) *Attribute:* Confirmed service indication

Invoke ID

This attribute shall identify the Transaction Object within the VCR.

VCR ID

This attribute shall identify the VCR on which the Transaction Object is used.

Confirmed service indication

Service identifier and argument of the pending service.

3.3.3.2 State Machine

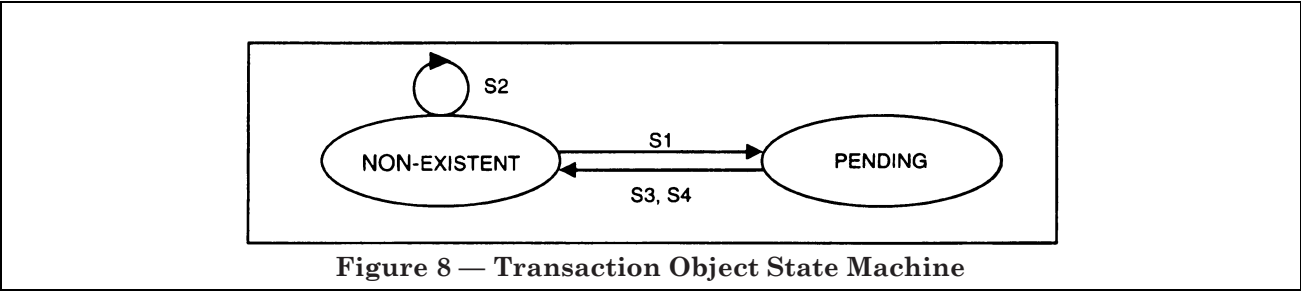
State Machine Description

NON-EXISTENT

The Transaction Object does not exist.

PENDING

A confirmed service indication primitive has been received and the corresponding response primitive has not yet been sent.



State Transitions

#	Current State	Events	Actions	Next State
S1	NON_EXISTENT	Confirmed service.ind && OSCR < Act MaxRCC	(no actions taken)	PENDING
S2	NON_EXISTENT	Confirmed service.ind && OSCR = Act MaxRCC	(no actions taken)	NON_EXISTENT
S3	PENDING	Confirmed service.rsp	(no actions taken)	NON_EXISTENT
S4	PENDING	Abort.ind	(no actions taken)	NON_EXISTENT

3.3.4 Context Management Services

The Context Management Services comprise:

- Initiate
- Abort
- Reject.

3.3.4.1 Initiate

This service shall be used to establish a connection between two communication partners and to exchange information regarding the supported services, the supported options, the maximum PDU length and the current version of the OD.

When the type of a VCR is such that FMS Initiate PDU is not exchanged, the request is locally confirmed and no information on the communication partner is available in confirmation.

Table 16 — Initiate

Parameter Name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
Version OD Calling	M	M(=)		
Profile Number Calling	M	M(=)		
Access Protection Supported Calling	M	M(=)		
Password Calling	M	M(=)		
Access Groups Calling	M	M(=)		
Destination DL-Address	C			
Result(+)			S	S(=)
Version OD Called			M	M(=)
Profile Number Called			M	M(=)
Access Protection Supported Called			M	M(=)
Password Called			M	M(=)
Access Groups Called			M	M(=)
Result(-)			S	S(=)
Error Code			M	M(=)
Max FMS PDU Sending Called				M
Max FMS PDU Receiving Called				M
FMS Features Supported Called				M

Argument

The argument shall convey the service specific parameters of the service request.

Version OD Calling

This parameter shall specify the version of the client's Object Dictionary.

Profile Number Calling

This parameter shall specify the Profile Number of the client.

Access Protection Supported Calling

This parameter shall contain the Access Protection Supported attribute of the client's OD object description.

Password Calling

This parameter shall specify the password to be used for the access to all objects of the server on this VCR. If access with password is not to be used on this VCR, the password parameter shall have the value 0.

Access Groups Calling

This parameter shall specify the client's membership in specific access groups. The membership applies for the access to all objects of the server on this VCR.

Destination DL-Address

This parameter exists only when the type of corresponding VCR is QUB and the remote partner has FREE endpoint. It gives the remote address to which the connection shall be established.

Result(+)

The Result(+) parameter shall indicate that the requested service was executed successfully.

Version OD Called

This parameter shall specify the version of the server's Object Dictionary.

Profile Number Called

This parameter shall specify the Profile Number of the server.

Access Protection Supported Called

This parameter shall contain the Access Protection Supported attribute of the server's OD object description.

Password Called

This parameter shall specify the password to be used for access to all objects of the client on this VCR. If access with password is not to be used on this VCR, the password parameter shall have the value 0.

Access Groups Called

This parameter shall specify the server's membership in specific access groups. The membership shall be used for the access to all objects of the client on this VCR.

Result(-)

The Result(-) parameter shall indicate that the service request failed.

Error Code

This parameter shall provide the reason for the failure.

- Max FMS PDU Size Insufficient. The maximum PDU length is not sufficient for the communication.
- Feature Not Supported The requested service or option is not supported by the server.
- User Initiate Denied. The FMS user refuses to establish the connection.
- Version OD incompatible. The versions of the Object Dictionaries (Source OD and Remote OD) of the communication partners are not compatible.
- Password Error. There is already a VCR established with the same password.
- Profile Number incompatible. The client's profile is not supported by the server.
- Other. Reason other than any of those identified above.

Max FMS PDU Sending Called

This parameter shall specify the maximum length of the FMS PDU to be sent that may be handled on this VCR.

It shall be transmitted by the called FMS and shall only be part of the Initiate.cnf primitive.

Max FMS PDU Receiving Called

This parameter shall specify the maximum length of the FMS PDU to be received that may be handled on this VCR.

It shall be transmitted by the called FMS and shall only be part of the Initiate.cnf primitive.

FMS Features Supported Called

This parameter shall identify the optional FMS services and the options supported by the server (see VCR List).

It shall be transmitted by the called FMS and shall only be part of the Initiate.cnf primitive.

3.3.4.2 Abort

This service shall be used to release an existing VCR between two communication partners. Both client and server may release the connection.

Table 17 — Abort

Parameter Name	.req	.ind
Argument	M	M
Locally Generated		M
Abort Identifier	M	M(=)
Reason Code	M	M(=)
Abort Detail	U	C

Argument

The argument shall convey the service specific parameters of the service request.

Locally Generated

This parameter shall indicate whether the abort was generated locally or by the communication partner. The value false is not allowed, if the Abort Identifier parameter has the value FMS and the Reason Code parameter has the value VCR Error.

Abort Identifier

This parameter shall indicate, where the reason for the abort has been detected. Possible values are:

FMS USER
FMS
FAS
DLL

Reason Code

This parameter shall specify the reason for the abort.

If the Abort Identifier parameter has the value USER, the following values shall apply:

- 0 <=> Disconnect. The connection is released by the FMS user.
- 1 <=> Version OD incompatible. The versions of the Object Dictionaries (Source OD and Remote OD) of the communication partners are not compatible.
- 2 <=> Password Error. There is already a VCR established with the same password.
- 3 <=> Profile Number incompatible. The server's profile is not supported by the client.
- 4 <=> Limited Services Permitted. The VFD is in the Logical Status LIMITED-SERVICES-PERMITTED.

If the Abort Identifier parameter has the value FMS, the following values shall apply:

- 0 <=> VCR Error. Faulty VCR
- 1 <=> User Error. Improper, unknown or faulty service primitive received from the FMS user
- 2 <=> FMS PDU Error. Unknown or faulty FMS PDU received from the FAS
- 3 <=> Connection State Conflict FAS. Improper FAS service primitive
- 4 <=> FAS Error. Unknown or faulty FAS service primitive

- 5 <=> PDU Size. PDU length exceeds maximum PDU length
- 6 <=> Feature Not Supported. SERVICE-REQ_PDU received from the FAS and service or option not supported as a server (see FMS Features Supported attribute in the VCR)
- 7 <=> Invoke ID Error Response. Confirmed service.rsp received from the FMS user and Invoke ID does not exist or CONFIRMED_SERVICE-RSP_PDU received from the FAS and Invoke ID does not exist
- 8 <=> Max Services Overflow. CONFIRMED_SERVICE-REQ_PDU received from the FAS and Outstanding Services Counter Receiving $\geq$ Max Outstanding Services Receiving
- 9 <=> Connection State Conflict FMS INITIATE-REQ_PDU received from the FAS
- 10 <=> Service Error. The service in the response does not match the service in the indication or the service in the confirmation does not match the service in the request.
- 11 <=> Invoke ID Error Request. CONFIRMED-SERVICE_REQ_PDU received from the FAS and Invoke ID already exists.

If the Abort Identifier parameter has the value FAS or DLL, the Reason Code parameter shall be supplied by the FAS.

Abort Detail

This parameter contains additional information about the abort reason (max. 16 octets). In case of error reports from the application, the meaning is defined in the profile.

3.3.4.3 Reject

The Reject service is used by the FMS to reject an improper PDU.

Table 18 — Reject

Parameter Name	.ind
Argument	
Detected Here	M
Original Invoke ID	C
Reject PDU Type	M
Reject Code	M

Argument

The argument shall convey the service specific parameters of the service request.

Detected Here

This parameter shall indicate if the error has been detected locally (true). The value false is allowed only if the Reject PDU Type parameter has the value Confirmed-Response-PDU and the Reject Code parameter has the value PDU Size.

Original Invoke ID

The original Invoke ID of the rejected PDU.

Reject PDU Type

This parameter shall indicate the type of the rejected PDU. The permissible values are as follows:

- 1 <=> Confirmed-Request-PDU
- 2 <=> Confirmed-Response-PDU
- 3 <=> Unconfirmed-PDU
- 4 <=> Unknown type of PDU

Reject Code

This parameter shall specify the reason for the reject.

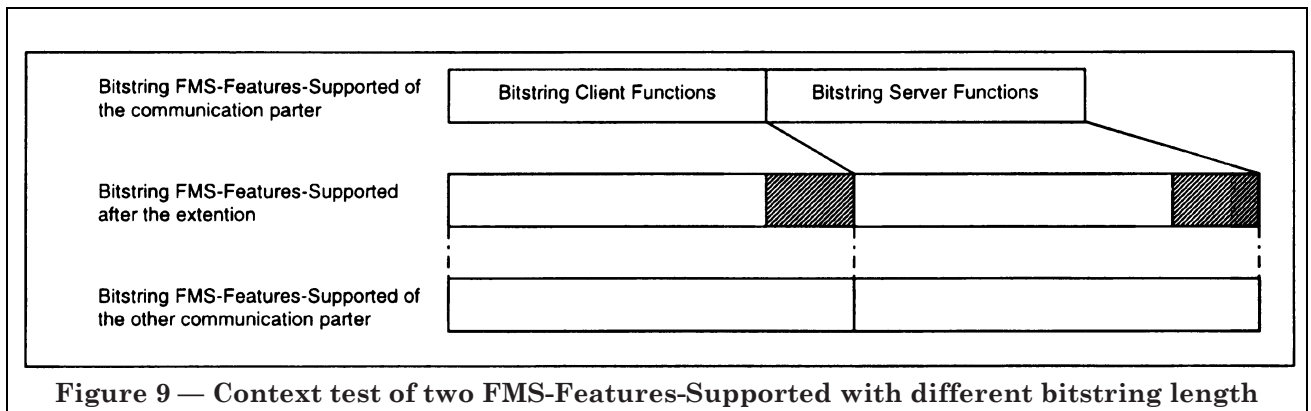
- 1 <=> Invoke-ID-Exists. Confirmed service.req received from the FMS user and Invoke ID already exists
- 2 <=> Max-Services-Overflow. Confirmed service.req received from the FMS user and OSCS = ActualMaxSCC

- ### 3.3.5 Tests at Connection Establishment

Upon receipt of an INITIATE-REQ_PDU the called FMS shall check the FMS Context of the communication partner (remote Context), if it is compatible with its own FMS Context (local Context) as defined for this connection in this VCR. The local FMS Context is assumed to be correct. The compatibility of the local with the remote context is defined by the following table. In this table, meaningless fields are left blank (e.g. the combination of local Context “Max FMS PDU Sending” with remote Context “FMS Features Supported”). Those combinations shall not be checked.

Remote Content		Local Content					
		Max FMS PDU		FMS Feature Supported			
		Sending	Receiving	.req [n]		.ind [n]	
0	1			0	1		
Max FMS PDU Sending			≥				
Max FMS PDU Receiving		≤					
FMS Features Supported	.req [n] 0					X	X
	.req [n] 1					—	X
	.ind [n] 0			X	—		
	.ind [n] 1			X	X		
Explanation:							
.req 0:	Feature is not used as Client						
.req 1:	Feature is used as Client						
.ind 0:	Feature is not supported as Server						
.ind 1:	Feature is supported as Server						
≤ :	local value smaller than or equal remote value						
≥ :	local value larger than or equal remote value						
X :	compatible						
— :	not compatible (error case)						
[n] :	0 to (23+8x); without extension ==> x= 0; with extension ==> x= 1,2, ...						

26



MaxRCC and MaxSCC of initiating side FMS are shown in INITIATE-REQ_PDU. Upon reception of this PDU, the called FMS shall store the smaller one of MaxRCC in the PDU and its own MaxSCC to its ActualMaxSCC. It shall also store the smaller one of MaxSCC in the PDU and its own MaxRCC to its ActualMaxRCC. The called FMS shall return the resultant ActualMaxRCC and ActualMaxSCC in the MaxRCC and MaxSCC fields of INITIATE-RSP_PDU. Upon reception of this PDU, the initiating FMS shall store them in its own Act MaxSCC and ActualMaxRCC, respectively. If the ActualMaxRCC is greater than MaxRCC or ActualMaxSCC is greater than MaxSCC, the initiating FMS shall abort this connection.

3.3.5.2 Tests at the FMS User

- Password Test. If the FMS user supports access with password, it shall check if the password is unambiguous. That means, if there is a password, it shall be different from the passwords of all other VCRs.
- If the password is not unambiguous and the FMS user is the server of the Initiate service, it shall issue an Initiate.rsp primitive with the Result(–) parameter and with Error Code “Password Error”.
- If the password is not unambiguous and the FMS user is the client of the Initiate service, it shall release the connection with the Abort service, with Reason Code “Password Error”.
- Test of Version OD. If the FMS user has a remote OD for this connection, it may check if the received parameter “Version OD” is compatible with the attribute “Version OD” of the associated remote OD.
- If they are incompatible and the FMS user is the server of the Initiate service, it may issue an Initiate.rsp primitive with the Result(–) parameter and with Error Code “Version OD incompatible”.
- If they are incompatible and the FMS user is the client of the Initiate service, it may release the connection with the Abort service, with Reason Code “Version OD incompatible”.
- Test of Profile Number. The FMS user may check if the received parameter “Profile Number” is compatible with the attribute “Profile Number” of the VFD.
- If they are incompatible and the FMS user is the server of the Initiate service, it may issue an Initiate.rsp primitive with the Result(–) parameter and with Error Code “Profile Number incompatible”.
- If they are incompatible and the FMS user is the client of the Initiate service, it may release the connection with the Abort service, with Reason Code “Profile Number incompatible”.

3.3.6 State Machine for VCRs

3.3.6.1 State Machine Description

CONNECTION-NOT-ESTABLISHED

The connection is not established. Only the service primitives Initiate.req, ASC.ind, Abort.req and ABT.ind are allowed. All other services shall be rejected with the Abort service.

CONNECTION-ESTABLISHING (CALLING)

The local FMS user wishes to establish the connection. Only the service primitives ASC.cnf(+), ASC.cnf(–), Abort.req and ABT.ind are allowed. All other services shall be rejected with the Abort service. This state is abbreviated as CON-ESTABLISHING-CALLING.

CONNECTION-ESTABLISHING (CALLED)

The remote FMS user wishes to establish the connection. Only the service primitives Initiate.rsp(+), Initiate.rsp(-), Abort.req and ABT.ind are allowed. All other services shall be rejected with the Abort service. This state is abbreviated as CON-ESTABLISHING-CALLED.

CONNECTION-ESTABLISHED

The VCR is established. The service primitives Initiate.req, Initiate.rsp(+), Initiate.rsp(-) are not allowed and shall be rejected with the Abort service. The following actions shall be taken to reset a VCR (Reset VCR):

- Set attribute “Outstanding Services Counter Sending” and attribute “Outstanding Services Counter Receiving” of the VCR (dynamic part) to 0
- Set state of the VCR to “CONNECTION-NOT-ESTABLISHED”

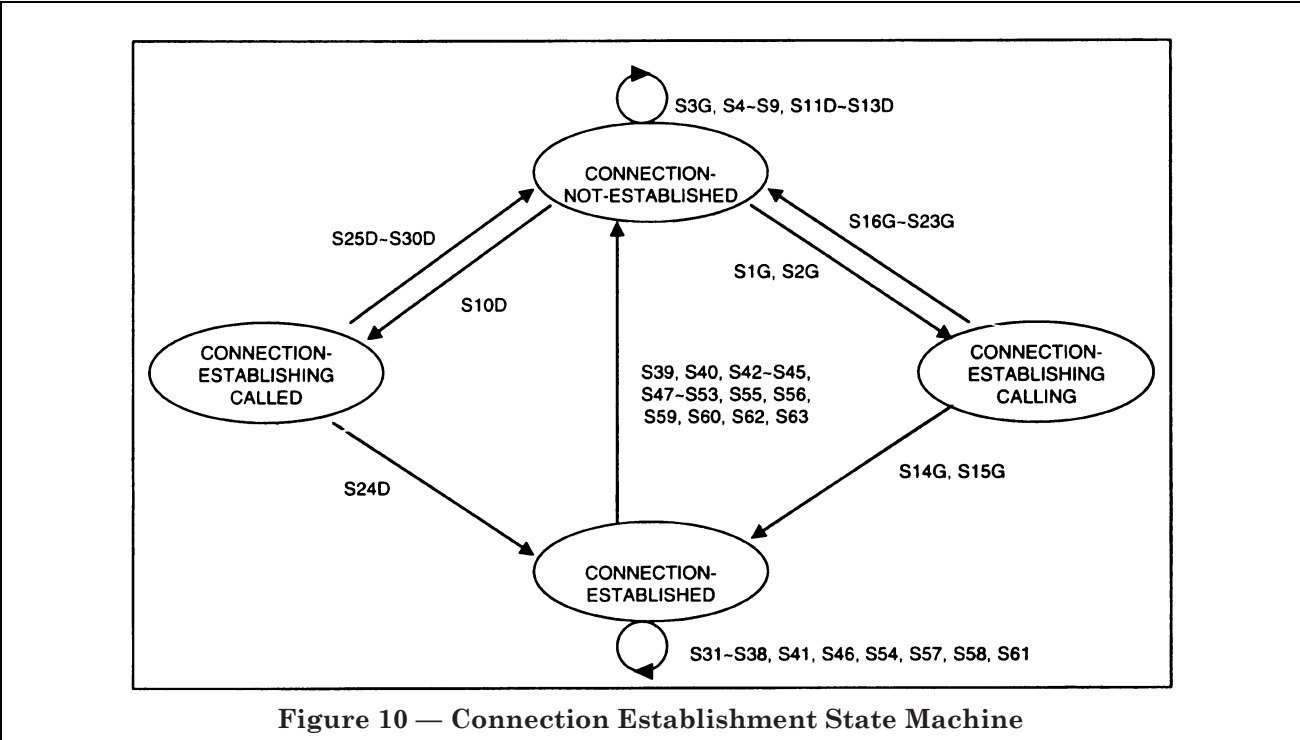


Figure 10 — Connection Establishment State Machine

Transition numbers with “G” are for calling side while transitions numbers with “D” are for called side.

3.3.6.2 Connection Establishment State Transitions

#	Current State	Events	Actions	Next State
S1G	CONNECTION-NOT-ESTABLISHED	Initiate.req && VCR is valid && VCR Type = QUB	ASC.req{ Data := Initiate-RequestPDU }	CONNECTION-ESTABLISHING-CALLING
S2G	CONNECTION-NOT-ESTABLISHED	Initiate.req && VCR is valid && VCR Type <> QUB	ASC.req{ Data := NULL }	CONNECTION-ESTABLISHING-CALLING
S3G	CONNECTION-NOT-ESTABLISHED	Initiate.req && VCR is not valid	Abort.ind{ AbortIdentifier := FMS, ReasonCode := VCR error }	CONNECTION-NOT-ESTABLISHED
S4	CONNECTION-NOT-ESTABLISHED	Abort.req	(no actions taken)	CONNECTION-NOT-ESTABLISHED

#	Current State	Events	Actions	Next State
S5	CONNECTION-NOT-ESTABLISHED	FMS service.primitive other than Initiate & Abort	Abort.ind{ AbortIdentifier := FMS, ReasonCode := User error }	CONNECTION-NOT-ESTABLISHED
S6	CONNECTION-NOT-ESTABLISHED	ABT.ind	(no actions taken)	CONNECTION-NOT-ESTABLISHED
S7	CONNECTION-NOT-ESTABLISHED	Faulty or unknown FAS service primitive	ABT.req{ AbortIdentifier := FMS, ReasonCode := FAS error }	CONNECTION-NOT-ESTABLISHED
S8	CONNECTION-NOT-ESTABLISHED	FAS service primitive other than ASC.ind and ABT.ind	ABT.req{ AbortIdentifier := FMS, ReasonCode := Connection state conflict FAS }	CONNECTION-NOT-ESTABLISHED
S9	CONNECTION-NOT-ESTABLISHED	ASC.ind && Data = not allowed, unknown or faulty FMS PDU	ABT.req{ AbortIdentifier := FMS, ReasonCode := FMS PDU error }	CONNECTION-NOT-ESTABLISHED
S10D	CONNECTION-NOT-ESTABLISHED	ASC.ind && Data Initiate-RequestPDU && VCR is valid && FMS context test is positive	Initiate.ind{ }	CONNECTION-ESTABLISHING-CALLED
S11D	CONNECTION-NOT-ESTABLISHED	ASC.ind && Data = Initiate-RequestPDU && VCR is invalid	ABT.req{ AbortIdentifier := FMS, ReasonCode := VCR error }	CONNECTION-NOT-ESTABLISHED
S12D	CONNECTION-NOT-ESTABLISHED	ASC.ind && Data = Initiate-RequestPDU && VCR is valid && FMS context test is positive && Max FMS PDU length test is negative	ASC.rsp(-){ Data := Initiate-ErrorPDU{ ErrorCode := max-fms-pdu-size-insufficient } }	CONNECTION-NOT-ESTABLISHED
S13D	CONNECTION-NOT-ESTABLISHED	ASC.ind && Data = Initiate-RequestPDU && VCR is valid && FMS context test is positive && Max FMS PDU length test is positive && FmsFeaturesSupported test is negative	ASC.rsp(-){ Data := Initiate-ErrorPDU{ ErrorCode := feature-not-supported } }	CONNECTION-NOT-ESTABLISHED
S14G	CONNECTION-ESTABLISHING-CALLING	ASC.cnf(+) && Data = Initiate-ResponsePDU && VCR Type = QUB	Initiate.cnf(+){ }	CONNECTION-ESTABLISHED
S15G	CONNECTION-ESTABLISHING-CALLING	ASC.cnf(+) && Data = NULL && VCR Type = QUU or BNU	Initiate.cnf(+){ }	CONNECTION-ESTABLISHED
S16G	CONNECTION-ESTABLISHING-CALLING	ASC.cnf(-) && Data = Initiate-ErrorPDU && VCR Type = QUB	Initiate.cnf(-){ ErrorCode := Error code in ASC.cnf }, reset VCR	CONNECTION-NOT-ESTABLISHED

#	Current State	Events	Actions	Next State
S17G	CONNECTION-ESTABLISHING-CALLING	ASC.cnf(-) && Data = NULL && VCR Type = QUU or BNU	Initiate.cnf(-){ ErrorCode := Error code in ASC.cnf }, reset VCR	CONNECTION-NOT-ESTABLISHED
S18G	CONNECTION-ESTABLISHING-CALLING	ABT.ind	Abort.ind{ AbortIdentifier := AbortIdentifier of ABT.ind, ReasonCode := ReasonCode of ABT.ind }, reset VCR	CONNECTION-NOT-ESTABLISHED
S19G	CONNECTION-ESTABLISHING-CALLING	Abort.req	ABT.req{ AbortIdentifier := AbortIdentifier of Abort.req, ReasonCode := ReasonCode of Abort.req }, reset VCR	CONNECTION-NOT-ESTABLISHED
S20G	CONNECTION-ESTABLISHING-CALLING	Faulty or unknown FAS service primitive	ABT.req{ AbortIdentifier := FMS, ReasonCode := FAS error }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := FAS error }, reset VCR	CONNECTION-NOT-ESTABLISHED
S21G	CONNECTION-ESTABLISHING-CALLING	FAS service primitive other than ASC.cnf and ABT.ind	ABT.req{ AbortIdentifier := FMS, ReasonCode := Connection state conflict FAS }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := Connection state conflict FAS }, reset VCR	CONNECTION-NOT-ESTABLISHED
S22G	CONNECTION-ESTABLISHING-CALLING	ASC.cnf && Data = not allowed, unknown or faulty FMS PDU	ABT.req{ AbortIdentifier := FMS, ReasonCode := FMS PDU error }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := FMS PDU error }, reset VCR	CONNECTION-NOT-ESTABLISHED
S23G	CONNECTION-ESTABLISHING-CALLING	FMS service.primitive other than Initiate.rsp and Abort.req	ABT.req{ AbortIdentifier := FMS, ReasonCode := User error }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := User error }, reset VCR	CONNECTION-NOT-ESTABLISHED
S24D	CONNECTION-ESTABLISHING-CALLED	Initiate.rsp(+)	ASC.rsp(+){ Data = Initiate-ResponsePDU }	CONNECTION-ESTABLISHED

#	Current State	Events	Actions	Next State
S25D	CONNECTION-ESTABLISHING-CALLED	Initiate.rsp(-)	ASC.rsp(-){ Data = Initiate-ErrorPDU{ ErrorCode = ErrorCode of Initiate.rsp } }, reset VCR	CONNECTION-NOT-ESTABLISHED
S26D	CONNECTION-ESTABLISHING-CALLED	Abort.req	ABT.req{ AbortIdentifier := AbortIdentifier of Abort.req, ReasonCode := ReasonCode of Abort.req }, reset VCR	CONNECTION-NOT-ESTABLISHED
S27D	CONNECTION-ESTABLISHING-CALLED	ABT.ind	Abort.ind{ AbortIdentifier := AbortIdentifier of ABT.ind, ReasonCode := ReasonCode of ABT.ind }, reset VCR	CONNECTION-NOT-ESTABLISHED
S28D	CONNECTION-ESTABLISHING-CALLED	Faulty or unknown FAS service primitive	ABT.req{ AbortIdentifier := FMS, ReasonCode := FAS error }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := FAS error }, reset VCR	CONNECTION-NOT-ESTABLISHED
S29D	CONNECTION-ESTABLISHING-CALLED	FAS service primitive other ABT.ind	ABT.req{ AbortIdentifier := FMS, ReasonCode := Connection state conflict FAS }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := Connection state conflict FAS }, reset VCR	CONNECTION-NOT-ESTABLISHED
S30D	CONNECTION-ESTABLISHING-CALLED	FMS service.primitive other than Initiate.rsp and Abort.req	ABT.req{ AbortIdentifier := FMS, ReasonCode := User error }, Abort.ind{ AbortIdentifier := FMS, ReasonCode := User error }, reset VCR	CONNECTION-NOT-ESTABLISHED
S31	CONNECTION-ESTABLISHED	ConfirmedService.req && OSCS < ActualMaxSCC && InvokeID not existent && PDU length ≤ Max FMS PDU sending && FmsFeaturesSupported test (client) positive	DTC.req{ Data := Confirmed-RequestPDU }, OSCS := OSCS + 1	CONNECTION-ESTABLISHED

#	Current State	Events	Actions	Next State
S32	CONNECTION-ESTABLISHED	ConfirmedService.req && OSCS = ActualMaxSCC	Reject.ind{ DetectedHere := true, Original InvokeID := InvokeID in ConfirmedService.req, RejectPDUType := Confirmed-RequestPDU, RejectCode := Max-services-overflow }	CONNECTION-ESTABLISHED
S33	CONNECTION-ESTABLISHED	ConfirmedService.req && OSCS < ActualMaxSCC && InvokeID already existent	Reject.ind{ DetectedHere := true, Original InvokeID := InvokeID in ConfirmedService.req, RejectPDUType := Confirmed-RequestPDU, RejectCode := Invoke-ID-exists }	CONNECTION-ESTABLISHED
S34	CONNECTION-ESTABLISHED	ConfirmedService.req && OSCS < ActualMaxSCC && InvokeID not existent && PDU length > Max FMS PDU sending	Reject.ind{ DetectedHere := true, Original InvokeID := InvokeID in ConfirmedService.req, RejectPDUType := Confirmed-RequestPDU, RejectCode := PDU-size }	CONNECTION-ESTABLISHED
S35	CONNECTION-ESTABLISHED	ConfirmedService.req && OSCS < ActualMaxSCC && InvokeID not existent && PDU length ≤ Max FMS PDU sending && FmsFeaturesSupported test (client) negative	Reject.ind{ DetectedHere := true, Original InvokeID := InvokeID in ConfirmedService.req, RejectPDUType := Confirmed-RequestPDU, RejectCode := Feature-not-supported- connection-oriented }	CONNECTION-ESTABLISHED
S36	CONNECTION-ESTABLISHED	UnconfirmedService.req && PDU length ≤ Max FMS PDU sending && FmsFeaturesSupported test (client) positive	DTU.req{ Data := Unconfirmed-PDU }	CONNECTION-ESTABLISHED
S37	CONNECTION-ESTABLISHED	UnconfirmedService.req && PDU length > Max FMS PDU sending	Reject.ind{ DetectedHere := true, Original InvokeID := InvokeID in UnconfirmedService.req, RejectPDUType := Unconfirmed-PDU, RejectCode := PDU-size }	CONNECTION-ESTABLISHED
S38	CONNECTION-ESTABLISHED	UnconfirmedService.req && PDU length ≤ Max FMS PDU sending && FmsFeaturesSupported test (client) negative	Reject.ind{ DetectedHere := true, Original InvokeID := InvokeID in UnconfirmedService.req, RejectPDUType := Unconfirmed-PDU, RejectCode := Feature-not-supported- connection-oriented }	CONNECTION-ESTABLISHED