

Fieldbus specification

Part 5. Fieldbus message specification

ICS 35.100: 35.200

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 June 1996

© BSI 1996

Amendments issued since publication

Amd. No.	Date	Text affected

The following BSI reference relates to the work on this standard:
Committee reference AMT/7

ISBN 0 580 25851 3

Contents

	Page
Committees responsible	Inside front cover
Foreword	ii
Text of FF-94-870	1

Foreword

This Part of DD 236 has been prepared by Technical Committee, AMT/7.

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	System architecture	FF-94-800	Revision 1.0
Part 2	Network management	FF-94-801	Revision 1.0
Part 3	Data link layer services subset	FF-94-821	Revision 1.0
Part 4	Data link protocol specification	FF-94-822	Revision 1.0
Part 5	Fieldbus message specification	FF-94-870	Revision 1.0
Part 6	Fieldbus access sublayer (services and protocol)	FF-94-875	Revision 1.0

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 *BSI News*.

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.

Fieldbus Foundation

Fieldbus Specification

Fieldbus Message Specification

DOCUMENT: FF-94-870
REVISION : PS 1.0
DATE: August 28, 1995

© 1995 by the Fieldbus Foundation. All rights reserved.

Foreword

This document provides a specification for the Application Layer protocol to be implemented by systems covered by the Fieldbus Foundation.

Open Issue List		
No.	Description of Issue	Affected Documents

Resolution List		
No.	Description of Resolution	Affected Sections

Comments to be Addressed in the Final Specifications

1. The encoding rules should specify that ASN.1 tagged productions are encoded in order, and only once, for each SEQUENCE or CHOICE in which they are defined.
2. The FMS Abort Reason Codes should have values assigned that are passed to FAS as shown in the FMS Connection Establishment State Transitions.

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. 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 The Fieldbus Foundation 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.

Table of Contents

1. Scope	3
2. Architecture	3
3. FMS Models and Services	3
3.1 VFD Support	3
3.1.1 Model Description	3
3.1.2 The VFD Object	4
3.1.3 VFD Support Services	5
3.2 Object Dictionary (OD) Management	7
3.2.1 Model Description	7
3.2.2 Structure of an OD	8
3.2.3 Object Description in the OD	10
3.2.4 OD Object Description	10
3.2.5 The OD Object	12
3.2.6 OD Services	12
3.2.7 State Machine	16
3.3 Context Management	19
3.3.1 Model Description	19
3.3.2 The VCR List	19
3.3.3 The Transaction Object	21
3.3.4 Context Management Services	22
3.3.5 Tests at Connection Establishment	25
3.3.6 State Machine for VCRs	26
3.4 Domain Management	32
3.4.1 Model Description	32
3.4.2 The Domain Object	32
3.4.3 Generic Download Services	34
3.4.4 Download Services	36
3.4.5 Upload Services	38
3.4.6 State Machine for GenericDownload/Download	41
3.4.7 State Machine for Upload	43
3.4.8 Examples	45
3.5 Program Invocation Management	46
3.5.1 Model Description	46
3.5.2 The Program Invocation (PI) Object	46
3.5.3 Program Invocation Services	48
3.5.4 Program Invocation State Machine	53
3.6 Variable Access	57
3.6.1 Model Description	57
3.6.2 Variable Access Objects	57
3.6.3 Variable Access Services	65
3.7 Event Management	73
3.7.1 Model Description	73
3.7.2 The Event Object	74
3.7.3 Event Management Services	74
3.7.4 Event Notification State Machine	77
3.7.5 Examples	78
4. Interface between FMS and FAS	80
4.1 Overview of Services	80
4.2 Mapping of FMS Services on to FAS Services	80
5. Operating Behavior of FMS	80
5.1 Start of FMS	80
5.2 Conditions of Readiness for Operation	80
5.3 FMS Readiness for Operation	80
6. Structured Parameters ErrorType and Type Description	81
6.1 Parameter Error Type	81
6.1.2 Meaning of Error Class and Error Code	81
6.1.3 Meaning of the remaining Parameters	82
6.2 Parameter Type Description	82
7. Tables	83

7.1 List of Object Codes	83
7.2 List of Standard Data Types	83
7.3 List of Object Attributes and Service Parameters	84
7.4 List of Services	85
8. Conformance	86
8.1 Implementation and System (PICS Part 1)	86
8.2 Supported Services (PICS Part 2)	86
8.3 FMS Parameters and Options (PICS Part 3)	86
8.4 Local Implementation Values (PICS Part 4)	87
9. Coding	88
9.1 General	88
9.2 Coding Rules	88
9.3 Structure of the Identification Information	88
9.3.1 Coding of Application Data	89
9.3.2 Coding of Structure Information	94
10. Syntax Description	96
10.1 The FMS PDU	96
10.1.1 Description of the fixed PDU Part	96
10.1.2 ConfirmedServiceRequest	96
10.1.3 ConfirmedServiceResponse	97
10.1.4 ServiceError	98
10.1.5 VFD Support	101
10.1.6 OD Management	102
10.1.7 Context Management	103
10.1.8 Domain Management	104
10.1.9 Program Invocation Management	106
10.1.10 Variable Access	108
10.1.11 Event Management	111
10.1.12 List of packed fields	112
10.1.13 Detailed Coding Examples (Informative)	113

1. Scope

The Fieldbus message specification (FMS) is based on a series of object oriented models that provide distributed applications with the functionality 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 FF System Architecture provided an overview of the Fieldbus architecture, application relationships and the concepts of objects, services, addressing and protection mechanisms used in Fieldbus.

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

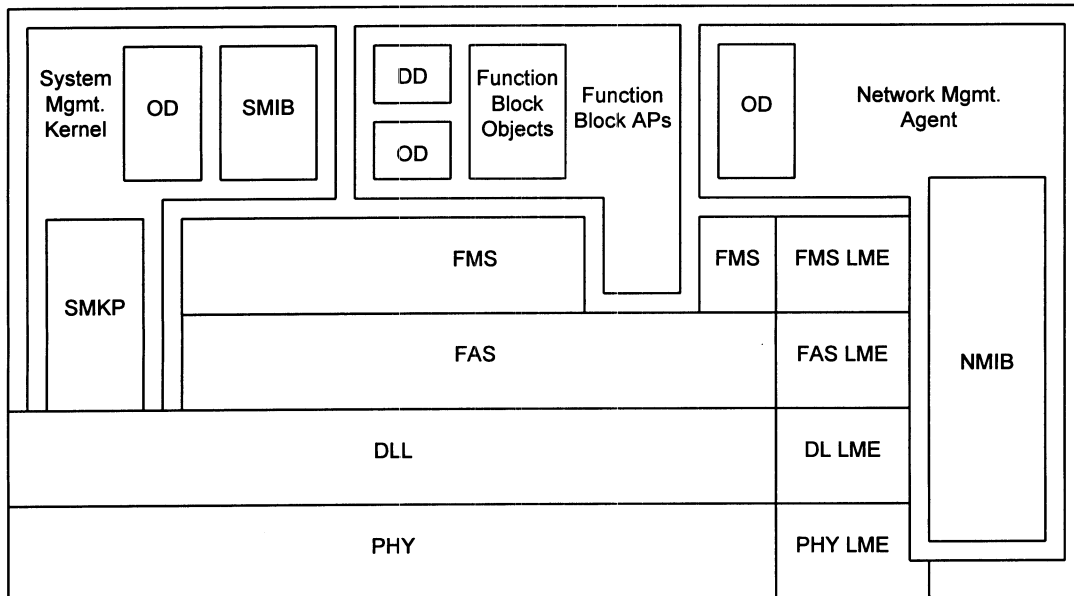


Figure 1 – General Fieldbus Foundation Architecture

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.

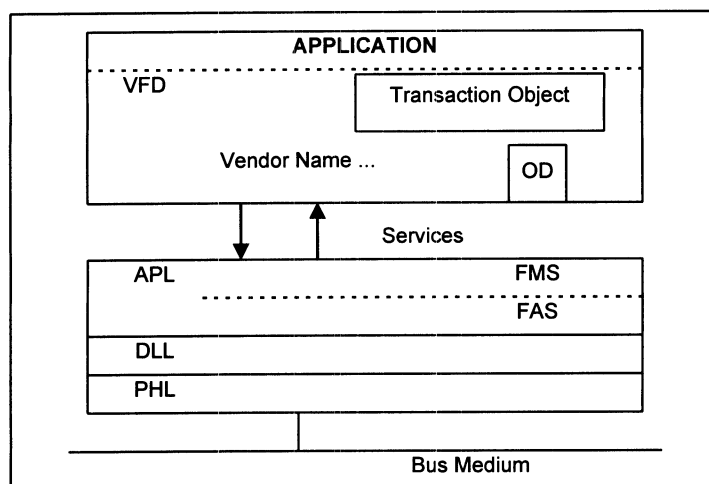


Figure 2 – VFD Model

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 .

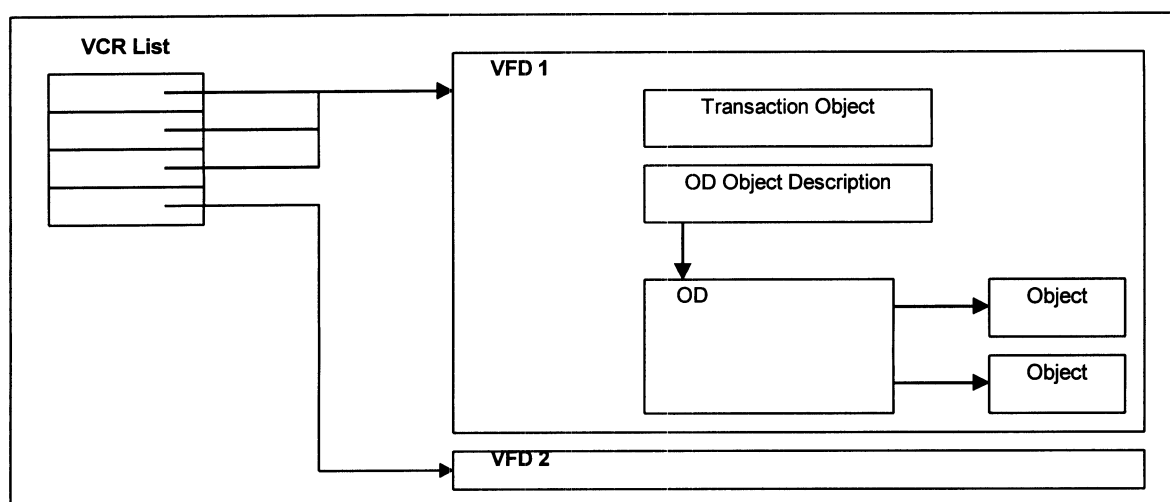


Figure 3 – Abstract Model of an Automation System (VFD)

The following VFD Support Services are defined:

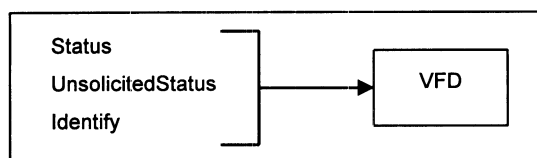


Figure 4 – VFD Support Services

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 functionality of the device.

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

Ready for communication

All services may be used normally.

Limited number of services

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

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.

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.

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(=)

Table 1 – Status

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 functionality 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.

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

Table 2 – UnsolicitedStatus

Argument:

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

LogicalStatus:

This parameter contains information about the state of the communication functionality 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

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(=)

Table 3 – Identify

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

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 a unique index of type unsigned16.

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

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

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.

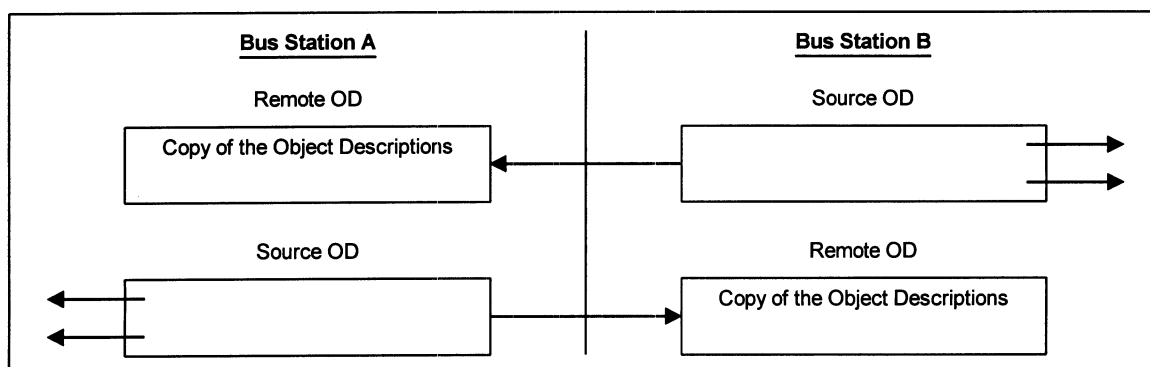


Figure 5 – Source OD/Remote OD

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.

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)

Table 4 – Structure of the Object Dictionary

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.

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

Table 5 – Structure of the Static List of Types

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.

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

Table 6 – Structure of the Static Object Dictionary

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.

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

Table 7 – Structure of the Dynamic List of Variable Lists

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.

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

Table 8 – Structure of the Dynamic List of 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 $\Leftrightarrow$ no modifications in the OD are permitted

true $\Leftrightarrow$ modifications in the OD are permitted

NameLength:

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

0 $\Leftrightarrow$ 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 $S\text{-OD-Length} = \text{Firstindex-S-OD} + S\text{-OD-Length} - 1$ (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 $\Leftrightarrow$ 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 $DV\text{-OD-Length} = \text{Firstindex-DV-OD} + DV\text{-OD-Length} - 1$

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 $\Leftrightarrow$ 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 $DP\text{-OD-Length} = \text{Firstindex-DP-OD} + DP\text{-OD-Length} - 1$

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:

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

Table 9 – Empty Object Dictionary

3.2.5 The OD Object

3.2.5.1 Attributes

Class:	OD
1. (m)	<i>Key Attribute:</i> implicit by VFD
2. (m)	<i>Attribute:</i> OD object description
3. (m)	<i>Attribute:</i> 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.

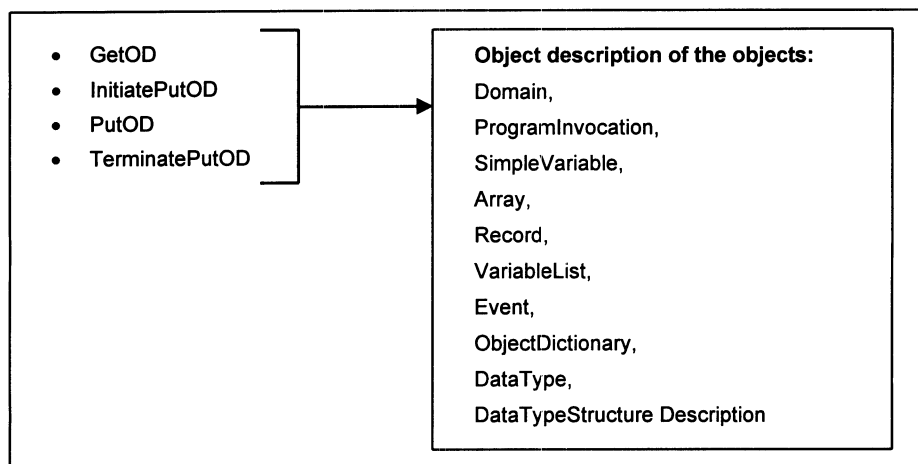
List of object description:

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.

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(=)

Table 10 – GetOD

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:

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

AccessSpecification:

This parameter states which object description is accessed.

Index:

Index of the object description.

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 shall 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.

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

Table 11 – InitiatePutOD

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 <=> newload, 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.

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(=)

Table 12 – PutOD

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.

Index	Object Code
246	Null Object

Table 13 – Empty Object Description (Example)

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.

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

Table 14 – TerminatePutOD

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.

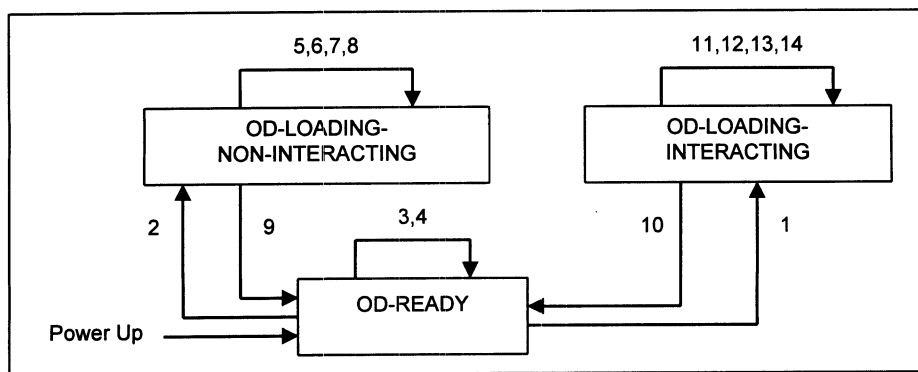


Figure 6 – State Machine

3.2.7.2 State Transitions

Current State Event Exit Condition => Action	Transition	Next State
OD-READY InitiatePutOD.ind \consequence >0 => InitiatePutOD.rsp(+)	1	OD-LOADING-INTERACTING
OD-READY InitiatePutOD.ind \consequence =0 => InitiatePutOD.rsp(+)	2	OD-LOADING-NON-INTERACTING
OD-READY PutOD.ind => PutOD.rsp(-) object state conflict	3	OD-READY
OD-READY TerminatePutOD.ind => TerminatePutOD.rsp(-) object state conflict	4	OD-READY
OD-LOADING-NON-INTERACTING InitiatePutOD.ind \consequence >0 => InitiatePutOD.rsp(-) object state conflict	5	OD-LOADING-NON-INTERACTING
OD-LOADING-NON-INTERACTING InitiatePutOD.ind \consequence =0 => InitiatePutOD.rsp(-) object state conflict	6	OD-LOADING-NON-INTERACTING
OD-LOADING-NON-INTERACTING PutOD.ind => PutOD.rsp(+)	7	OD-LOADING-NON-INTERACTING
OD-LOADING-NON-INTERACTING TerminatePutOD.ind \OD not usable => TerminatePutOD.rsp(-) operational problem	8	OD-LOADING-NON-INTERACTING
OD-LOADING-NON-INTERACTING TerminatePutOD.ind \OD okay => TerminatePutOD.rsp(+)	9	OD-READY
OD-LOADING-INTERACTING TerminatePutOD.ind \OD okay => TerminatePutOD.rsp(+)	10	OD-READY
OD-LOADING-INTERACTING InitiatePutOD.ind \consequence =0 => InitiatePutOD.rsp(-) object state conflict	11	OD-LOADING-INTERACTING

Current State Event \\Exit Condition => Action	Transition	Next State
OD-LOADING-INTERACTING PutOD.ind => PutOD.rsp(+)	12	OD-LOADING-INTERACTING
OD-LOADING-INTERACTING TerminatePutOD.ind \\OD not usable => TerminatePutOD.rsp(-) operational problem	13	OD-LOADING-INTERACTING
OD-LOADING-INTERACTING InitiatePutOD.ind \\consequence >0 => InitiatePutOD.rsp(+)	14	OD-LOADING-INTERACTING

3.2.7.3 Example of a PutOD

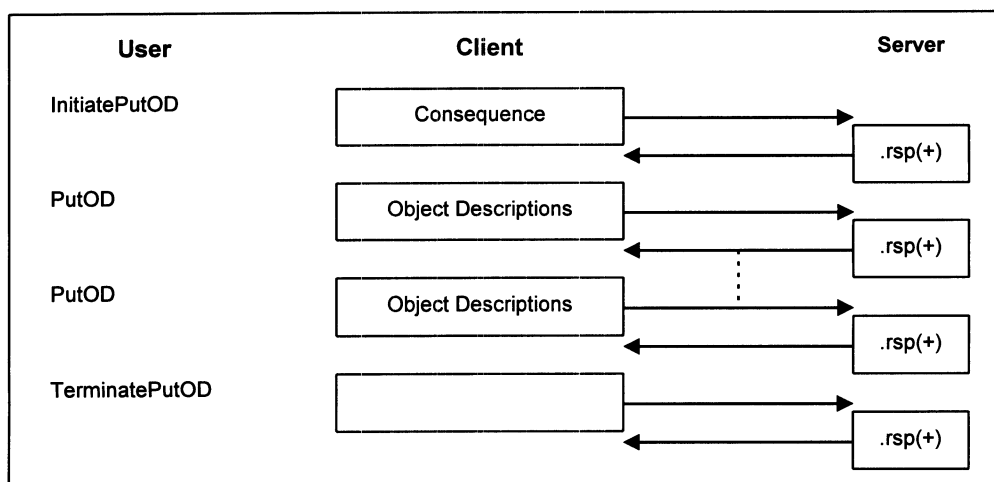


Figure 7 – Example of a PutOD Sequence

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

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 bitstring of the FMS Features Supported is of the length specified by FMS Features Supported Length and consists of two bit strings of the same length. The first bitstring identifies services and features as a client and the second bitstring 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 bitstring. For the optional feature there are two bits, one in the first bitstring, 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 bitstring.. All bits within a byte, which are not specified, have the value 0.

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, ...				

Table 15 – Attribute FMS Features Supported

Max Outstanding Services Calling (Max SCC):

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 Called (Max RCC):

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 Calling (OSCC):

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

Outstanding Services Counter Called (OSCS):

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

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 Calling(Act. Max SCC):

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.

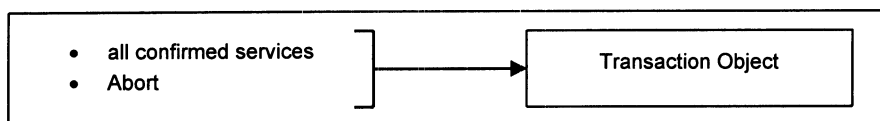
Actual Outstanding Services Called(Act. Max RCC):

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 Called 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.

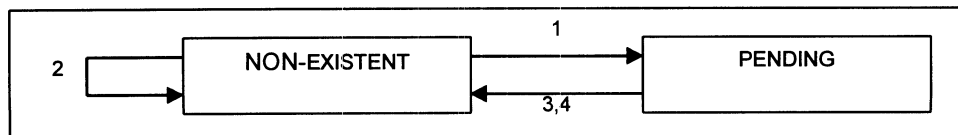


Figure 8 – Transaction Object State Machine

State Transitions

1. Confirmed service.ind (Number of Transaction Objects < Act Max RCC)
2. Confirmed service.ind (Number of Transaction Objects = Act Max RCC)

3. Confirmed service.rsp
4. Abort.ind

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.

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

Table 16 – Initiate

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.

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

Table 17 – Abort

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:

ABT_RC1 <=> Disconnect. The connection is released by the FMS user.
ABT_RC2 <=> Version OD incompatible. The versions of the Object Dictionaries (Source OD and Remote OD) of the communication partners are not compatible.
ABT_RC3 <=> Password Error. There is already a VCR established with the same password.
ABT_RC4 <=> Profile Number incompatible. The server's profile is not supported by the client.
ABT_RC5 <=> 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:

ABT_RC1 <=> VCR Error. Faulty VCR
ABT_RC2 <=> User Error. Improper, unknown or faulty service primitive received from the FMS user
ABT_RC3 <=> FMS PDU Error. Unknown or faulty FMS PDU received from the FAS
ABT_RC4 <=> Connection State Conflict FAS. Improper FAS service primitive
ABT_RC5 <=> FAS Error . Unknown or faulty FAS service primitive
ABT_RC6 <=> PDU Size. PDU length exceeds maximum PDU length
ABT_RC7 <=> 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)
ABT_RC8 <=> Invoke ID Error Response. Confirmed service.rsp received from the FMS user and Invoke ID does not exist or CONFIRMED-SERVICE_RES_PDU received from the FAS and Invoke ID does not exist
ABT_RC9 <=> Max Services Overflow. CONFIRMED-SERVICE_REQ_PDU received from the FAS and Outstanding Services Counter Server _ Outstanding Services Server
ABT_RC10 <=> Connection State Conflict FMS. INITIATE_REQ_PDU received from the FAS
ABT_RC11 <=> 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.
ABT_RC12 <=> 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.

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

Table 18 – Reject

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 Out-standing Services Counter Client_Outstanding Services Client
- 3 <=> Feature-Not-Supported-Connection-Oriented. Service.req received from the FMS user and service or option not supported as a client (see FMS Features Supported attribute in the VCR)
- 4 <=> Feature-Not-Supported-Connectionless. Unconfirmed service.req received from the FMS user and service or option not supported as a client (see FMS Features Supported attribute in the VCR) or confirmed service.req received from the FMS user
- 5 <=> PDU-Size. PDU length exceeds maximum PDU length
- 6 <=> User-Error-Connectionless. Improper or faulty service primitive received from the FMS user
- 0 <=> Other. Reason other than any of those identified above.

3.3.5 Tests at Connection Establishment**3.3.5.1 Context Test in FMS**

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, ...							

Table 19 – Compatibility of the local FMS Context to the remote FMS Context

The length of the FMS-Features-Supported is not limited. So it can happen, that the bitstring length of the FMS-Features-Supported is different between the bitstring length of the FMS-Features-Supported from the communication partners. To check the FMS context it is necessary to approximate the both bitstring length, which shall be compared. The rule is, that the bitstring with the shorter length has to be extended with bits of the value 0, until both bitstrings are equal. Note, that the bitstring of the FMS Features Supported consists of a bitstring for the primitive of client functions and a bitstring for the primitive of server functions. The extension has to be symmetrical between the bitstring for the primitive of client functions and the bitstring for the primitive of server functions.

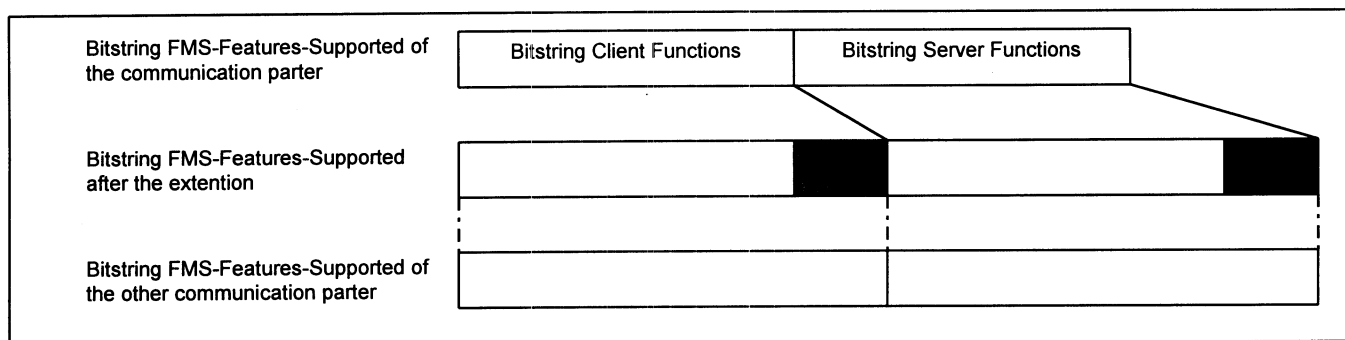


Figure 9 – Context test of two FMS-Features-Supported with different bitstring length

Max RCC and Max SCC of initiating side FMS are shown in INITIATE_REQ_PDU. Upon reception of this PDU, the called FMS shall store the smaller one of Max RCC in the PDU and its own Max SCC to its Act. Max SCC. It shall also store the smaller one of Max SCC in the PDU and its own Max RCC to its Act. Max RCC. The called FMS shall return the resultant Act. Max RCC and Act. Max SCC in the Max RCC and Max SCC fields of INITIATE_RSP_PDU. Upon reception of this PDU, the initiating FMS shall store them in its own Act Max SCC and Act. Max RCC, respectively. If the Act. Max RCC is greater than Max RCC or Act. Max SCC is greater than Max SCC, 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.response 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.response 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.response 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 Calling" and attribute "Outstanding Services Counter Called" of the VCR (dynamic part) to 0
- Set state of the VCR to "CONNECTION-NOT-ESTABLISHED"

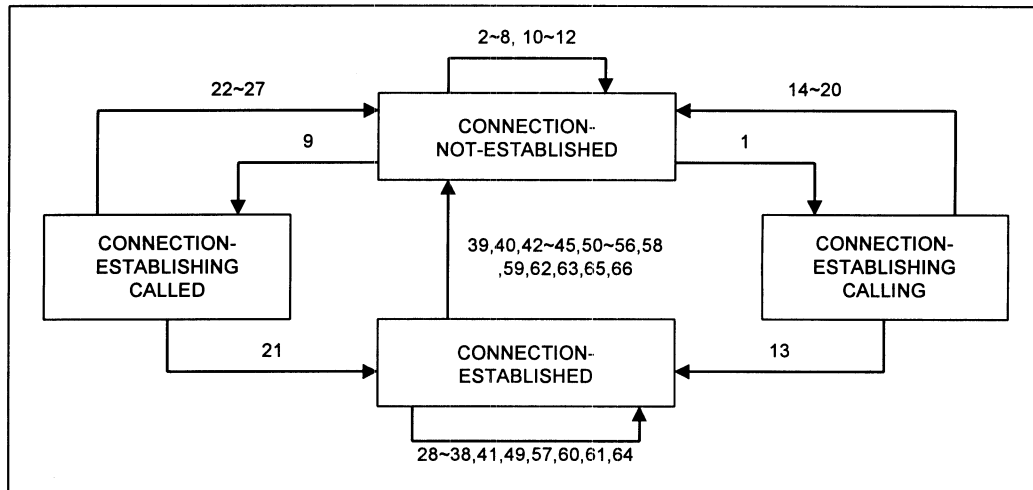


Figure 10 – Connection Establishment State Machine

3.3.6.2 Connection Establishment State Transitions

State Transitions at Connection Establishment/Release:		
Current State	Transition	Next State
Event	Exit Condition	=> Action Taken
CONNECTION-NOT-ESTABLISHED	1	CON-ESTABLISHING-CALLING
Initiate.req from FMS user received		\FMS VCR entry valid If VCR Type = QUB => send INITIATE_REQ_PDU, (ASC.req) to FAS else => send local ASC.req (without FMS PDU) to FAS
CONNECTION-NOT-ESTABLISHED	2	CONNECTION-NOT-ESTABLISHED
Initiate.req from FMS user received		\VCR entry invalid => Abort.ind to FMS user <reason code = ABT_RC1>
CONNECTION-NOT-ESTABLISHED	3	CONNECTION-NOT-ESTABLISHED
Abort.req received from FMS user		=> ignore
CONNECTION-NOT-ESTABLISHED	4	CONNECTION-NOT-ESTABLISHED
not allowed, unknown or faulty service primitive received from FMS user		=> Abort.ind to FMS user <reason code = ABT_RC2>
CONNECTION-NOT-ESTABLISHED	5	CONNECTION-NOT-ESTABLISHED
ABT.ind received from FAS		=> ignore
CONNECTION-NOT-ESTABLISHED	6	CONNECTION-NOT-ESTABLISHED
faulty or unknown service primitive received from FAS		=> ABT.req to FAS <reason code = ABT_RC5>
CONNECTION-NOT-ESTABLISHED	7	CONNECTION-NOT-ESTABLISHED
not allowed FAS service primitive received from FAS		=> ABT.req to FAS <reason code = ABT_RC4>
CONNECTION-NOT-ESTABLISHED	8	CONNECTION-NOT-ESTABLISHED
not allowed, unknown or faulty FMS PDU (ASC.ind) received		=> ABT.req to FAS <reason code = ABT_RC3>
CONNECTION-NOT-ESTABLISHED	9	CON-ESTABLISHING-CALLED
INITIATE_REQ_PDU(ASC.ind from FAS)		\VCR entry is valid AND FMS context test is positive => Initiate.ind to FMS user
CONNECTION-NOT-ESTABLISHED	10	CONNECTION-NOT-ESTABLISHED
INITIATE_REQ_PDU(ASC.ind from FAS)		\VCR entry is invalid => ABT.req to FAS <reason code = ABT_RC1>
CONNECTION-NOT-ESTABLISHED	11	CONNECTION-NOT-ESTABLISHED
INITIATE_REQ_PDU (ASC.ind) from FAS		\VCR entry is valid AND max FMS PDU length test is negative => send INITIATE_ERROR_PDU (ASC.rsp(-)) to FAS <error code = 1>

State Transitions at Connection Establishment/Release:		
Current State	Transition	Next State
Event	Exit Condition	=> Action Taken
CONNECTION-NOT-ESTABLISHED	12	CONNECTION-NOT-ESTABLISHED
INITIATE_REQ_PDU (ASC.ind) from FAS		\VCR entry is valid AND max FMS PDU length test is positive AND features supported test is negative => send INITIATE_ERROR_PDU (ASC.rsp(-)) to FAS <error code = 2>
CON-ESTABLISHING-CALLING	13	CONNECTION-ESTABLISHED
INITIATE_RES_PDU (ASC.cnf(+)) from FAS		\VCR Type = QUB => Initiate.cnf(+) to FMS user
CON-ESTABLISHING-CALLING	13A	CONNECTION-ESTABLISHED
_No FMS PDU (ASC.cnf(+)) from FAS		\VCR Type = QUU or BNU => Initiate.cnf(+) to FMS user
CON-ESTABLISHING-CALLING	14	CONNECTION-NOT-ESTABLISHED
INITIATE_ERROR_PDU (ASC.cnf(-)) from FAS		\VCR Type = QUB => Initiate.cnf(-) to FMS user reset VCR
CON-ESTABLISHING-CALLING	14 A	CONNECTION-NOT-ESTABLISHED
_No FMS_PDU (ASC.cnf(-)) from FAS		\VCR Type = QUU or BNU => Initiate.cnf(-) to FMS user reset VCR
CON-ESTABLISHING-CALLING	15	CONNECTION-NOT-ESTABLISHED
ABT.ind received from FAS		=> Abort.ind to FMS user <reason code out of ABT.ind> reset VCR
CON-ESTABLISHING-CALLING	16	CONNECTION-NOT-ESTABLISHED
Abort.req received from FMS user		=> ABT.req to FAS <reason code as given by user> reset VCR
CON-ESTABLISHING-CALLING	17	CONNECTION-NOT-ESTABLISHED
faulty or unknown service primitive received from FAS		=> ABT.req to FAS <reason code = ABT_RC5> Abort.ind to FMS user <reason code = ABT_RC5> reset VCR
CON-ESTABLISHING-CALLING	18	CONNECTION-NOT-ESTABLISHED
not allowed FAS service primitive received from FAS		=> ABT.req to FAS <reason code = ABT_RC4> Abort.ind to FMS user <reason code = ABT_RC4> reset VCR
CON-ESTABLISHING-CALLING	19	CONNECTION-NOT-ESTABLISHED
not allowed, unknown or faulty FMS PDU (ASC.cnf) received from FAS		=> ABT.req to FAS <reason code = ABT_RC3> Abort.ind to FMS user <reason code = ABT_RC3> reset VCR

Current State Event \\Exit Condition => Action Taken	Transition	Next State
CON-ESTABLISHING-CALLING not allowed, unknown or faulty service primitive received from FMS user => ABT.req to FAS <reason code = ABT_RC2> Abort.ind to FMS user <reason code = ABT_RC2> reset VCR	20	CONNECTION-NOT-ESTABLISHED
CON-ESTABLISHING-CALLED Initiate.rsp(+) received from FMS user => send INITIATE_RES_PDU (ASC.rsp(+)) to FAS	21	CONNECTION-ESTABLISHED
CON-ESTABLISHING-CALLED Initiate.rsp(-) received from FMS user => send INITIATE_ERROR_PDU (ASC.rsp(-)) to FAS reset VCR	22	CONNECTION-NOT-ESTABLISHED
CON-ESTABLISHING-CALLED Abort.req received from FMS user => ABT.req to FAS <reason code as given by user> reset VCR	23	CONNECTION-NOT-ESTABLISHED
DN-ESTABLISHING-CALLED ABT.ind received from FAS => Abort.ind to FMS user <reason code out of ABT.ind> reset VCR	24	CONNECTION-NOT-ESTABLISHED
DN-ESTABLISHING-CALLED faulty or unknown service primitive received from FAS => ABT.req to FAS <reason code = ABT_RC5> Abort.ind to FMS user <reason code = ABT_RC5> reset VCR	25	CONNECTION-NOT-ESTABLISHED
DN-ESTABLISHING-CALLED not allowed FAS service primitive received from FAS => ABT.req to FAS <reason code = ABT_RC4> Abort.ind to FMS user <reason code = ABT_RC4> reset VCR	26	CONNECTION-NOT-ESTABLISHED
DN-ESTABLISHING-CALLED not allowed, unknown or faulty service primitive received from FMS user => ABT.req to FAS <reason code = ABT_RC2> Abort.ind to FMS user <reason code = ABT_RC2> reset VCR	27	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED confirmed Service.req received from FMS user \\OSCC < Actual Max Outstanding Services Calling AND invoke ID not existent AND PDU length ≤ max FMS PDU sending AND features supported test (Client) positive => confirmed service_REQ_PDU with DTC.req to FAS OSCC := OSCC + 1	28	CONNECTION-ESTABLISHED
CONNECTION-ESTABLISHED confirmed Service.req received from FMS user \\OSCC = Actual Max Outstanding Services Calling => Reject.ind to FMS user <reject code = 2>	29	CONNECTION-ESTABLISHED

Current State Event \\Exit Condition => Action Taken	Transition	Next State
CONNECTION-ESTABLISHED confirmed Service.req received from FMS user \\OSCC < Max Outstanding Services Calling AND invoke ID already existent => Reject.ind to FMS user <reject code = 1>	30	CONNECTION-ESTABLISHED
CONNECTION-ESTABLISHED confirmed Service.req received from FMS user \\OSCC < Max Outstanding Services Calling AND invoke ID not existent AND PDU length > max FMS PDU sending => Reject.ind to FMS user <reject code = 5>	31	CONNECTION-ESTABLISHED
CONNECTION-ESTABLISHED confirmed Service.req received from FMS user \\OSCC < Max Outstanding Services Calling AND invoke ID not existent AND PDU length ≤ max FMS PDU sending AND features supported test (Client) negative => Reject.ind to FMS user <reject code = 3>	32	CONNECTION-ESTABLISHED
CONNECTION-ESTABLISHED unconfirmed Service.req received from FMS user \\PDU length ≤ max FMS PDU sending AND features supported test (Client) positive => unconfirmed Service_REQ_PDU (DTU.req) to FAS	36	CONNECTION-ESTABLISHED
CONNECTION-ESTABLISHED unconfirmed Service.req > received from FMS user \\PDU length > max FMS PDU sending => Reject.ind to FMS user <reject code = 5>	37	CONNECTION-ESTABLISHED
CONNECTION-ESTABLISHED unconfirmed Service.req received from FMS user \\PDU length ≤ max FMS PDU sending AND features supported test (Client) negative => Reject.ind to FMS user <reject code = 3>	38	CONNECTION-ESTABLISHED
CONNECTION-ESTABLISHED Abort.req received from FMS user => ABT.req to FAS <reason code as given by user> reset VCR	39	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED faulty, unknown or not allowed service primitive received from FMS user => ABT.req to FAS <reason code = ABT_RC2> Abort.ind to FMS user <reason code = ABT_RC2> reset VCR	40	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED confirmed Service_REQ_PDU (DTC.ind) from FAS \\PDU length ≤ max FMS PDU receiving AND OSCS < Max Outstanding Services Called AND invoke ID not existent AND features supported test (Server) positive => confirmed Service.ind to FMS user OSCS := OSCS + 1	41	CONNECTION-ESTABLISHED

Current State Event Exit Condition => Action Taken	Transition	Next State
CONNECTION-ESTABLISHED confirmed Service_REQ_PDU (DTC.ind) from FAS PDU length > max FMS PDU receiving => ABT.req to FAS <reason code = ABT_RC6> Abort.ind to FMS user <reason code = ABT_RC6> reset VCR	42	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED confirmed Service_REQ_PDU (DTC.ind) from FAS PDU length ≤ max FMS PDU receiving AND OSCS ≥ Max Outstanding Services Called => ABT.req to FAS <reason code = ABT_RC9> Abort.ind to FMS user <reason code = ABT_RC9> reset VCR	43	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED confirmed Service_REQ_PDU (DTC.ind) from FAS PDU length ≤ max FMS PDU receiving AND OSCS < Max Outstanding Services Called AND invoke ID already existent => ABT.req to FAS <reason code = ABT_RC12> Abort.ind to FMS <reason code = ABT_RC12> reset VCR	44	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED confirmed Service_REQ_PDU (DTC.ind) from FAS PDU length ≤ max FMS PDU receiving AND OSCS < Max Outstanding Services Called AND invoke ID not existent AND features supported test (Server) negative => ABT.req to FAS <reason code = ABT_RC7> Abort.ind to FMS user <reason code = ABT_RC7> reset VCR	45	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED unconfirmed Service_REQ_PDU (DTU.ind) from FAS PDU length ≤ max FMS PDU receiving AND features supported test (Server) positive => unconfirmed Service.ind to FMS user	49	CONNECTION-ESTABLISHED
CONNECTION-ESTABLISHED unconfirmed Service_REQ_PDU (DTU.ind) from FAS PDU length > max FMS PDU receiving => ABT.req to FAS <reason code = ABT_RC6> Abort.ind to FMS user <reason code = ABT_RC6> reset VCR	50	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED unconfirmed Service_REQ_PDU (DTU.ind) from FAS PDU length ≤ max FMS PDU receiving AND features supported test (Server) negative => ABT.req to FAS <reason code = ABT_RC7> Abort.ind to FMS user <reason code = ABT_RC7> reset VCR	51	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED ABT.ind from FAS received => Abort.ind to FMS user <reason code out of ABT.ind> reset VCR	52	CONNECTION-NOT-ESTABLISHED

Current State Event Exit Condition => Action Taken	Transition	Next State
CONNECTION-ESTABLISHED INITIATE_REQ_PDU (ASC.ind) from FAS => ABT.req to FAS <reason code = ABT_RC10> Abort.ind to FMS user <reason code = ABT_RC10> reset VCR	53	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED faulty or unknown service primitive from FAS received => ABT.req to FAS <reason code = ABT_RC5> Abort.ind to FMS user <reason code = ABT_RC5> reset VCR	54	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED not allowed FAS service primitive received from FAS => ABT.req to FAS <reason code = ABT_RC4> Abort.ind to FMS user <reason code = ABT_RC4> reset VCR	55	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED not allowed, unknown or faulty FMS PDU received from FAS => ABT.req to FAS <reason code = ABT_RC3> Abort.ind to FMS user <reason code = ABT_RC3> reset VCR	56	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED confirmed Service.rsp received from FMS user Invoke ID existent as server AND service out of .rsp identical with service out of .ind AND PDU length ≤ max FMS PDU sending => confirmed Service_RES_PDU with DTC.rsp to FAS OSCS := OSCS - 1	57	CONNECTION-ESTABLISHED
CONNECTION-ESTABLISHED confirmed Service.rsp received from FMS user Invoke ID not existent as server => ABT.req to FAS <reason code = ABT_RC8> Abort.ind to FMS user <reason code = ABT_RC8> reset VCR	58	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED confirmed Service.rsp received from FMS user Invoke ID existent as server AND service out of .rsp not identical with service out of .ind => ABT.req to FAS <reason code = ABT_RC11> Abort.ind to FMS user <reason code = ABT_RC11> reset VCR	59	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED confirmed Service.rsp received from FMS user Invoke ID existent as server AND service out of .rsp identical with Service out of .ind AND PDU length > max FMS PDU sending => REJECT_PDU with DTC.rsp to FAS <reject code = 5> OSCS := OSCS - 1	60	CONNECTION-ESTABLISHED

Current State	Transition	Next State
Event Exit Condition => Action Taken		
CONNECTION-ESTABLISHED confirmed Service_RES_PDU (DTC.cnf) from FAS PDU length ≤ max FMS PDU receiving AND invoke ID existent as client AND service out of .cnf identical with service out of .req => confirmed Service.cnf to FMS user OSCC := OSCC - 1	61	CONNECTION-ESTABLISHED
CONNECTION-ESTABLISHED confirmed Service_RES_PDU (DTC.cnf) from FAS PDU length > max FMS PDU receiving => ABT.req to FAS <reason code = ABT_RC6> Abort.ind to FMS user <reason code = ABT_RC6> reset VCR	62	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED confirmed Service_RES_PDU (DTC.cnf) from FAS PDU length ≤ max FMS PDU receiving AND invoke ID not existent as client => ABT.req to FAS <reason code = ABT_RC8> Abort.ind to FMS user <reason code = ABT_RC8> reset VCR	63	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED REJECT_PDU (DTC.cnf) from FAS Original invoke ID existent as client AND <reject code> = 5 => Reject.ind to FMS user <reject code = 5> OSCC := OSCC - 1	64	CONNECTION-ESTABLISHED
CONNECTION-ESTABLISHED REJECT_PDU (DTC.cnf) from FAS Original invoke ID not existent as client => ABT.req to FAS <reason code = ABT_RC8> Abort.ind to FMS user <reason code = ABT_RC8> reset VCR	65	CONNECTION-NOT-ESTABLISHED
CONNECTION-ESTABLISHED REJECT_PDU (DTC.cnf) from FAS Original invoke ID existent as client AND <reject code> unequal 5 => ABT.req to FAS <reason code = ABT_RC3> Abort.ind to FMS user <reason code = ABT_RC3> reset VCR	66	CONNECTION-NOT-ESTABLISHED