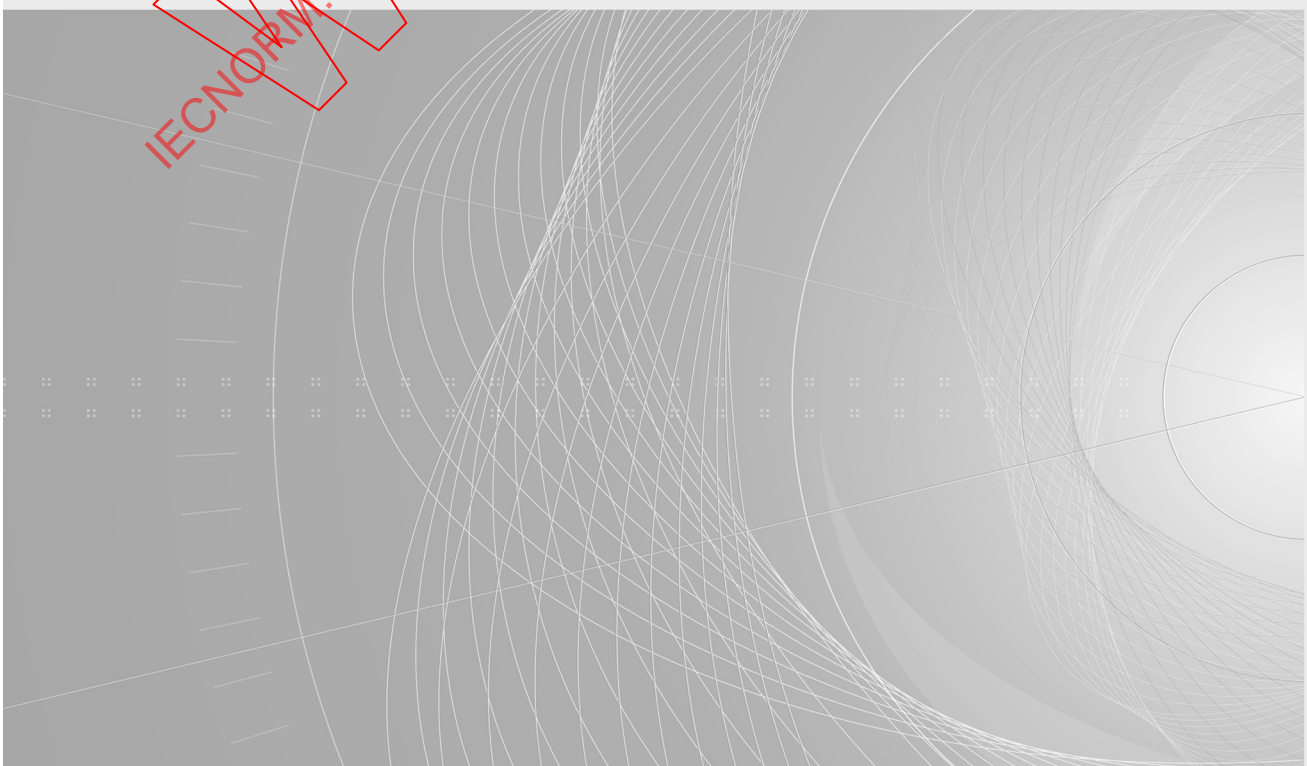
IEC 61158-6-20

Edition 2.0 2010-08

INTERNATIONAL STANDARD

Industrial communication networks – Fieldbus specifications – Part 6-20: Application layer protocol specification – Type 20 elements



IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub
The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub
Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Electropedia: www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv
If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC 61158-6-20

Edition 2.0 2010-08

INTERNATIONAL STANDARD

Industrial communication networks – Fieldbus specifications – Part 6-20: Application layer protocol specification – Type 20 elements

INTERNATIONAL ELECTROTECHNICAL COMMISSION

PRICE CODE XA

ICS 25.04.40; 35.100.70; 35.110

ISBN 978-2-88912-134-2

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-20: Application layer protocol specification – Type 20 elements

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

NOTE 1 Use of some of the associated protocol types is restricted by their intellectual-property-right holders. In all cases, the commitment to limited release of intellectual-property-rights made by the holders of those rights permits a particular data-link layer protocol type to be used with physical layer and application layer protocols in Type combinations as specified explicitly in the profile parts. Use of the various protocol types in other combinations may require permission from their respective intellectual-property-right holders.

International Standard IEC 61158-6-20 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This second edition cancels and replaces the first edition published in 2007. This edition constitutes a technical revision.

The main changes with respect to the previous edition are listed below:

a) revised Identify FAL PDU, see 5.3.2;

b) revised Read device variables with status FAL PDU, see 5.3.9;

The text of this standard is based on the following documents:

FDIS	Report on voting
65C/607/FDIS	65C/621/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with ISO/IEC Directives, Part 2.

A list of all parts of the IEC 61158 series, published under the general title *Industrial communication networks – Fieldbus specifications*, can be found on the IEC web site.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be:

- reconfirmed;
- withdrawn;
- replaced by a revised edition, or
- amended.

NOTE 2 The revision of this standard will be synchronized with the other parts of the IEC 61158 series.

INTRODUCTION

This part of IEC 61158 is one of a series produced to facilitate the interconnection of automation system components. It is related to other standards in the set as defined by the "three-layer" fieldbus reference model described in IEC/TR 61158-1.

The application protocol provides the application service by making use of the services available from the data-link or other immediately lower layer. The primary aim of this standard is to provide a set of rules for communication expressed in terms of the procedures to be carried out by peer application entities (AEs) at the time of communication. These rules for communication are intended to provide a sound basis for development in order to serve a variety of purposes:

- as a guide for implementors and designers;
- for use in the testing and procurement of equipment;
- as part of an agreement for the admittance of systems into the open systems environment;
- as a refinement to the understanding of time-critical communications within OSI.

This standard is concerned, in particular, with the communication and interworking of sensors, effectors and other automation devices. By using this standard together with other standards positioned within the OSI or fieldbus reference models, otherwise incompatible systems may work together in any combination.

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-20: Application layer protocol specification – Type 20 elements

1 Scope

1.1 General

The Fieldbus Application Layer (FAL) provides user programs with a means to access the fieldbus communication environment. In this respect, the FAL can be viewed as a "window between corresponding application programs."

This standard provides common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment and material specific to Type 20 fieldbus. The term "time-critical" is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This standard defines in an abstract way the externally visible behavior provided by the Type 20 of the fieldbus Application Layer in terms of

a) the abstract syntax defining the application layer protocol data units conveyed between communicating application entities,
b) the transfer syntax defining the application layer protocol data units conveyed between communicating application entities,
c) the application context state machine defining the application service behavior visible between communicating application entities; and
d) the application relationship state machines defining the communication behavior visible between communicating application entities; and.

The purpose of this standard is to define the protocol provided to define

a) the wire-representation of the service primitives defined in IEC 61158-5-20, and
b) the externally visible behavior associated with their transfer.

This standard specifies the protocol of the Type 20 IEC fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498) and the OSI Application Layer Structure (ISO/IEC 9545).

1.2 Specifications

The principal objective of this standard is to specify the syntax and behavior of the application layer protocol that conveys the application layer services defined in IEC 61158-5-20.

A secondary objective is to provide migration paths from previously-existing industrial communications protocols. It is this latter objective which gives rise to the diversity of protocols standardized in IEC 61158-6.

1.3 Conformance

This standard does not specify individual implementations or products, nor does it constrain the implementations of application layer entities within industrial automation systems. Conformance is achieved through implementation of this application layer protocol specification.

2 Normative references

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

IEC 60559, *Binary floating-point arithmetic for microprocessor systems*

IEC/TR 61158-1:2010[1], *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-5-20, *Industrial communication networks – Fieldbus specifications – Part 5-20: Application layer service definition – Type 20 elements*

ISO/IEC 7498-1, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*

ISO/IEC 8824-1, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ISO/IEC 8859-1, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*

ISO/IEC 9545, *Information technology – Open Systems Interconnection – Application Layer structure*

[1] To be published.

3 Terms, definitions, symbols, abbreviations and conventions

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

3.1 Terms and definitions from other ISO/IEC standards

3.1.1 Terms and definitions from ISO/IEC 7498-1

a) abstract syntax
b) application entity
c) application process
d) application protocol data unit
e) application service element
f) application entity invocation
g) application process invocation
h) application transaction
i) presentation context
j) real open system
k) transfer syntax

3.1.2 Terms and definitions from ISO/IEC 9545

a) application-association
b) application-context
c) application context name
d) application-entity-invocation
e) application-entity-type
f) application-process-invocation
g) application-process-type
h) application-service-element
i) application control service element

3.1.3 Terms and definitions from ISO/IEC 8824-1

a) object identifier
b) type
c) value
d) simple type
e) structured type
f) component type
g) tag
i) true
j) false
k) integer type
m) octet string type
n) null type
o) sequence type
p) sequence of type
q) choice type
r) tagged type
s) any type
t) module
u) production

3.1.4 Terms and definitions from ISO/IEC 8825

a) encoding (of a data value)
b) data value
c) identifier octets (the singular form is used in this standard)
d) length octet(s) (both singular and plural forms are used in this standard)
e) contents octets

3.2 IEC/TR 61158-1 terms

The following IEC/TR 61158-1 terms apply.

3.2.1
application
function or data structure for which data is consumed or produced

3.2.2
application layer interoperability
capability of application entities to perform coordinated and cooperative operations using the services of the FAL

3.2.3
application object
object class that manages and provides the run time exchange of messages across the network and within the network device

NOTE Multiple types of application object classes may be defined.

3.2.4
application process
part of a distributed application on a network, which is located on one device and unambiguously addressed

3.2.5
application process identifier
identifier that distinguishes among multiple application processes used in a device

3.2.6
application process object
component of an application process that is identifiable and accessible through an FAL application relationship

NOTE Application process object definitions are composed of a set of values for the attributes of their class (see the definition for Application Process Object Class Definition). Application process object definitions may be accessed remotely using the services of the FAL Object Management ASE. FAL Object Management services can be used to load or update object definitions, to read object definitions, and to dynamically create and delete application objects and their corresponding definitions.

3.2.7
application process object class
class of application process objects defined in terms of the set of their network-accessible attributes and services

3.2.8
application relationship
cooperative association between two or more application-entity-invocations for the purpose of exchange of information and coordination of their joint operation

NOTE This relationship is activated either by the exchange of application-protocol-data-units or as a result of pre-configuration activities

3.2.9
application relationship application service element
application-service-element that provides the exclusive means for establishing and terminating all application relationships

3.2.10
application relationship endpoint
context and behavior of an application relationship as seen and maintained by one of the application processes involved in the application relationship

NOTE Each application process involved in the application relationship maintains its own application relationship endpoint.

3.2.11
attribute
description of an externally visible characteristic or feature of an object

NOTE The attributes of an object contain information about variable portions of an object. Typically, they provide status information or govern the operation of an object. Attributes may also affect the behaviour of an object. Attributes are divided into class attributes and instance attributes.

3.2.12
behaviour
indication of how the object responds to particular events. Its description includes the relationship between attribute values and services

3.2.13
class
set of objects, all of which represent the same kind of system component

NOTE A class is a generalisation of the object; a template for defining variables and methods. All objects in a class are identical in form and behaviour, but usually contain different data in their attributes.

3.2.14
class attributes
attribute that is shared by all objects within the same class

3.2.15
class code
unique identifier assigned to each object class

3.2.16
class specific service
service defined by a particular object class to perform a required function which is not performed by a common service

NOTE A class specific object is unique to the object class which defines it.

3.2.17
client
(a) an object which uses the services of another (server) object to perform a task

(b) an initiator of a message to which a server reacts, such as the role of an AR endpoint in which it issues confirmed service request APDUs to a single AR endpoint acting as a server

3.2.18
conveyance path
unidirectional flow of APDUs across an application relationship

3.2.19
cyclic
term used to describe events which repeat in a regular and repetitive manner

3.2.20
dedicated AR
AR used directly by the FAL User. On Dedicated ARs, only the FAL Header and the user data are transferred

3.2.21
device
physical hardware connection to the link. A device may contain more than one node

3.2.22
device profile
collection of device dependent information and functionality providing consistency between similar devices of the same device type

3.2.23
endpoint
one of the communicating entities involved in a connection

3.2.24
error
discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition

3.2.25
error code
identification of a specific type of error within an error class

3.2.26
management information
network-accessible information that supports managing the operation of the fieldbus system, including the application layer

NOTE Managing includes functions such as controlling, monitoring, and diagnosing.

3.2.27
network
series of nodes connected by some type of communication medium

NOTE The connection paths between any pair of nodes can include repeaters, routers and gateways.

3.2.28
pre-defined AR endpoint
AR endpoint that is defined locally within a device without use of the create service

NOTE Pre-defined ARs that are not pre-established are established before being used.

3.2.29
pre-established AR endpoint
AR endpoint that is placed in an established state during configuration of the AEs that control its endpoints

3.2.30
server
a) role of an AREP in which it returns a confirmed service response APDU to the client that initiated the request

b) an object which provides services to another (client) object

3.2.31
service
operation or function than an object and/or object class performs upon request from another object and/or object class

NOTE A set of common services is defined and provisions for the definition of object-specific services are provided. Object-specific services are those which are defined by a particular object class to perform a required function which is not performed by a common service.

3.3 Type 20 fieldbus application-layer specific definitions

There are additional terms defined for this part.

3.3.1
analog channel
continuously varying electrical signal connecting a field device to the remainder of the data acquisition or control system. Some field devices support multiple analog channels (input or output)

NOTE Each analog channel transmits a single dynamic variable to or from the field device.

3.3.2
broadcast address
broadcast address is used by a master to send a command to all devices

NOTE The broadcast address is five octet long and has all zeros as the value.

3.3.3
busy
device is busy and cannot execute a command at the time; device indicates busy by returning response code 32 when allowed by the command specification and the requested command is not executed if a busy response is returned

3.3.4
device ID
serial number for a device

NOTE The manufacturer is required to assigned unique value for every device that has the identical values for Manufacturer ID and Device Type.

3.3.5
device type
manufacturer's type of a device, e.g. its product name

NOTE The value of this attribute is assigned by the manufacturer. Its value specifies the set of commands and data objects supported by the device. The manufacturer is required to assigned unique value to each type of the device.

3.3.6
device variable
uniquely defined data item within a Field Device that is always associated with cyclical process information and a device variable's value varies in response to changes and variations in the process to which the device is connected

3.3.7
dynamic variable
connection between a process and an analog channel

NOTE A device may contain primary, secondary, tertiary, and quaternary variables. These are collectively called the dynamic variables.

3.3.8
long tag
32-character restricted ISO Latin-1 string used to identify a field device

3.3.9
loop current
value measured by a milli-ammeter in series with the field device

NOTE The loop current is a near DC analog 4-20 mA signal used to communicate a single value between the control system and the field device. Voltage mode devices use "Volts DC" as their engineering units where "loop current" values are used.

3.3.10
manufacturer ID
string identifying the manufacturer that produced a device

NOTE A manufacturer is required to use the value assigned to it and is not permitted to use the value assigned to another manufacturer.

3.3.11
master
device that initiates communication activity by sending request PDUs to other devices

3.3.12
polling address
integer used to identify a device

NOTE The polling address is used to construct a one-octet address.

3.3.13
slave
device that initiates communication activity only after it receives a request PDU from a master device, and that is required to send a response to that request

3.3.14
tag
8-character ASCII string used to identify a field device

3.3.15
unique address
five-octet address of a device that uniquely identifies the device among all other devices that support this standard, formed by concatenating a manufacturer ID, a device type and a device ID

3.4 Abbreviations and symbols

AE	Application entity
AL	Application layer
ALP	Application layer protocol
APO	Application object
AP	Application process
APDU	Application protocol data unit
API	Application process Identifier
AR	Application relationship
AREP	Application relationship endpoint
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
Cnf	Confirmation
DL-	(as a prefix) data-link-
DLC	Data-link connection
DLCEP	Data-link connection endpoint
DLL	Data-link layer
DLM	Data-link-management
DLSAP	Data-link service access point
DLSDU	DL-service-data-unit
FAL	Fieldbus application layer

ID	Identifier
IEC	International Electrotechnical Commission
Ind	Indication
MSB	Most significant byte
LSB	Least significant byte
OSI	Open Systems Interconnect
Req	Request
Rsp	Response
SAP	Service access point
SDU	Service data unit
VFD	Virtual field device

3.5 Conventions

3.5.1 General concept

The FAL is defined as a set of object-oriented ASEs. Each ASE is specified in a separate subclause. Each ASE specification is composed of three parts: its class definitions, its services, and its protocol specification. The first two are contained in IEC 61158-5-20. The protocol specification for each of the ASEs is defined in this standard.

The class definitions define the attributes of the classes supported by each ASE. The attributes are accessible from instances of the class using the Management ASE services specified in IEC 61158-5-20. The service specification defines the services that are provided by the ASE.

This standard uses the descriptive conventions given in ISO/IEC 10731.

3.5.2 Conventions for class definitions

The data-link layer mapping definitions are described using templates. Each template consists of a list of attributes for the class. The general form of the template is defined in IEC/TR 61158-1.

3.5.3 Abstract syntax conventions

When the "optionalParametersMap" parameter is used, a bit number which corresponds to each OPTIONAL or DEFAULT production is given as a comment.

3.6 Conventions used in state machines

The state machines are described in Table 1.

Table 1 – Conventions used for state machines

#	Current state	Event / condition => action	Next state
Name of this transition	The current state to which this state transition applies	Events or conditions that trigger this state transaction. => The actions that are taken when the above events or conditions are met. The actions are always indented below events or conditions	The next state after the actions in this transition is taken

The conventions used in the state machines are as follows:

:= Value of an item on the left is replaced by value of an item on the right. If an item on the right is a parameter, it comes from the primitive shown as an input event.

xxx A parameter name.

Example:
Identifier := reason
means value of a 'reason' parameter is assigned to a parameter called 'Identifier.'

"xxx" Indicates fixed value.

Example:
Identifier := "abc"
means value "abc" is assigned to a parameter named 'Identifier.'

= A logical condition to indicate an item on the left is equal to an item on the right.

< A logical condition to indicate an item on the left is less than the item on the right.

> A logical condition to indicate an item on the left is greater than the item on the right.

<> A logical condition to indicate an item on the left is not equal to an item on the right.

&& Logical "AND"

|| Logical "OR"

This construct allows the execution of a sequence of actions in a loop within one transition. The loop is executed for all values from start_value to end_value.

Example:
for (Identifier := start_value to end_value)
actions
endfor

This construct allows the execution of alternative actions depending on some condition (which might be the value of some identifier or the outcome of a previous action) within one transition.

Example:
If (condition)
actions
else
actions
endif

Readers are strongly recommended to refer to the subclauses for the AREP attribute definitions, the local functions, and the FAL-PDU definitions to understand protocol machines. It is assumed that readers have sufficient knowledge of these definitions, and they are used without further explanations.

4 Abstract syntax

The abstract syntax of APDUs is combined with their transfer syntax and is specified in Clause 5.

5 Transfer syntax

5.1 General

The sending Application Layer prepares an APDU to transfer it to the receiving Application Layer. It uses the parameters from the service primitives to do so. There are several formats of the APDU:

- Request APDU from Master to Slave device,
- Normal response from Slave to Master device,

- Command error response from Slave to Master device, and
- Communication error response from Slave to Master device.

The format and coding rules for all APDUs are specified in this clause.

5.2 Common APDU structure

All APDU have a common structure as shown in Figure 1.

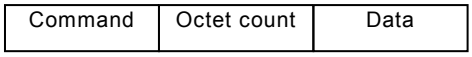


Figure 1 – APDU format

5.2.1 Command

The first octet of an APDU is Command. The Command is the Variable Index and it identifies the object in the Slave that has to be read or written. Its data type is Unsigned8.

5.2.2 Octet count

The second octet of an APDU represents the number of octets in Data field. The data type of this field is Unsigned8 and its value can be 0 to 255. If this field is zero, then Data field is null.

5.2.3 Data

This is the user data which is transferred between Application Layer and its user. The Application layer assembles it from the parameters of a service primitive or parses it into parameters of a service primitive. Its structure depends upon the type of APDU.

5.2.3.1 Master to slave request

Figure 1 shows the format of the application layer fields that are used for a Master request containing a single octet command number. For a Read service, Data field is either null or it is Variable Sub Index. For a Write service, Data field is the value that has to be written to the Slave objects identified by the Command.

5.2.3.2 Normal response from slave to master

Figure 2 shows the format of the application layer fields that are used for a response from Slave to Master without any error.

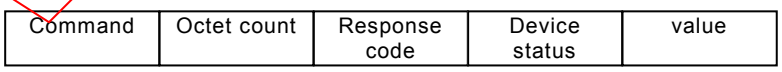


Figure 2 – Normal response from slave to master

The Slave sends this APDU in response to a request from Master. The value of Command field in this APDU is identical to the Command field in the corresponding request APDU. The Octet Count is number of octets in Value field plus two. Thus, its minimum value is 2.

Response Code field is Enumeration data type and it indicates the successful completion of the requested command as shown in Table 2. Some of the values depend upon the value of Command field. Each service-specific APDU structure in 5.3 shows the values assigned to that service. A device is required to use these values. The device can respond with an unused value as long it indicates a condition that is not defined in this standard.

Table 2 – Response code values

Value	Class	Definition
0	Success	Command (read or Write) was executed properly.
1 – 127	Warning	Command (Write) was executed with the deviation as described in response (e.g., a value was set to its nearest legal value).
1 – 127	Reserved	A code may be reserved for future use by this standard protocol. A device is not permitted to return reserved value.

Device Status field is Bit Field data type and it indicates the current operating status of the responding device as a whole and is not associated with the completion of any command, as shown in Table 3.

Table 3 – Device status values

Bit mask	Definition
0x80	Device Malfunction – the device detected a serious error or failure that compromises device operation.
0x40	Configuration Changed – an operation was performed that changed the device's configuration.
0x20	Cold Start – a power failure or Device Reset has occurred.
0x10	More Status Available – more status information is available via another Read Command.
0x08	Loop Current Fixed – the Loop Current is being held at a fixed value and is not responding to process variations.
0x04	Loop Current Saturated – the Loop Current has reached its upper (or lower) endpoint limit and cannot increase (or decrease) any further.
0x02	Non-Primary Variable Out of Limits – a Device Variable not mapped to the Primary Variable is beyond its operating limits.
0x01	Primary Variable Out of Limits – the Primary Variable is beyond its operating limit..

If the response is for a Read request, then Value field is used for the value of the objects that are read from Slave device. If the response is for a Write request, then this field is used for the value of the objects that were written in the Slave device. It is same as the value received in the request, unless the value received was not permitted and the server wrote a different value. In that case, it is the value written in the Slave device.

5.2.3.3 Command error response from slave to master

Figure 3 shows the format of the application layer fields that are used for a response from Slave to Master, when there is an error in executing the command. The Value field does not exist.

Command	Octet count (= 2)	Response code	Device status

Figure 3 – Command error response from slave to master

The responding device sends this APDU in response to a request from Master. The value of Command field in this APDU is identical to the Command field in the corresponding request APDU. The Octet Count field is set to two (2).

Response Code field is Enumeration data type and it indicates the reason for the error. The range of its value is 1 – 127. Some of the value depends upon the value of Command field. Each service-specific APDU structure in 5.3 shows the values assigned to that service. A device is required to use these values. The device can respond with an unused value as long it indicates a condition that is not defined in this standard. (The response code values are given in Table 4.)

Table 4 – Response code values

Value	Class	Definition
1 – 127	Error	Command execution was not properly completed and the Response Code indicates the reason (e.g., the device is in Write Protect mode).
1 – 127	Reserved	A code may be reserved for future use by this standard protocol. A device is not permitted to return reserved value.

Device Status field is Bit Field data type and it indicates the current operating status of the responding device as a whole and is not associated with the completion of any command, as shown in Table 3.

5.2.3.4 Communication error response from slave to master

Figure 4 shows the format of the application layer fields that are used for a response from Slave to Master, when there is a communication error in receiving the request DLPDU. The Value field does not exist.

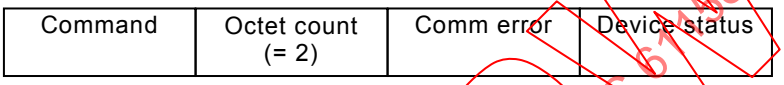


Figure 4 – Communication error response from slave to master

The responding device sends this APDU in response to a request from Master. The value of Command field in this APDU is identical to the Command field in the corresponding request APDU. The Octet Count field is set to two (2). Comm Error field is Bit Field data type and it indicates the reason for the error as shown in Table 5.

Table 5 – Communication error codes

Bit mask	Definition
0x80	1 - This bit is always set to '1' to indicate a communication error
0x40	Vertical Parity Error – The parity of one or more of the bytes received by the device was not odd.
0x20	Overrun Error – At least one byte of data in the receive buffer of the UART was overwritten before it was read (i.e., the slave did not process incoming byte fast enough).
0x10	Framing Error – The Stop Bit of one or more bytes received by the device was not detected by the UART (i.e. a mark or 1 was not detected when a Stop Bit should have occurred)
0x08	Longitudinal Parity Error – The Longitudinal Parity calculated by the device did not match the Check Byte at the end of the message.
0x04	Reserved, set to zero
0x02	Buffer Overflow – The message was too long for the receive buffer of the device.
0x01	Reserved, it is set to zero

5.3 Service-specific APDU structures

5.3.1 HCF enumerations

A number of data structures use enumerations which are controlled by the HART Communications Foundation (HCF). HCF maintains current lists of these enumerations.

5.3.2 Identify FAL PDU

5.3.2.1 Request primitive

The value of Command field can be either 0 or 11 or 21. The APDU format is shown in Table 6.

Table 6 – Identify request APDU

Command	Octet count	Data
0	0	Null
11	6	Tag (Packed ASCII)
21	32	Long Tag (Latin-1)

If the value of Command field is 0, then Value field is null. If the value of Command field is 11, then Value field contains six octet value of Tag parameter. If the value of Command field is 21, then Value field contains 32 octet value of Long Tag parameter.

5.3.2.2 Response primitive

The Value field of Identify response is shown in Table 7. The Response code values are shown in Table 8.

Table 7 – Identify response value field

Octet offset	Data type	Parameter name	Description
0	Unsigned8	None	is always 254
1	HCF enumeration	Expanded Device Type	is a 2-octet enumeration and indicates the manufacturer's type of the device. The manufacturer shall assign a unique value for every device type.
3	Unsigned8	Preamble Count	indicates the minimum number of Preambles to be sent with the request message from the Master to the responding device
4	Unsigned8	Command Rev	indicates the major revision level of the Protocol supported by the device
5	Unsigned8	Device Rev	describes the revision level of the device. The value of this parameter shall be defined by the manufacturer
6	Unsigned8	Soft Rev	indicates the revision level of the firmware in the device. The manufacturer shall increment the value of this parameter for every new release of the device's firmware
7	Bit Field	Hard Rev Phy Type	a) the five most significant bits of this octet represent the revision level of the device hardware. The manufacturer shall increment the value of this attribute for every major change of the device's hardware. It is not necessary to track individual hardware component changes. b) the least three significant bits of this octet represent the type of PhL signalling used by the device
8	Bit Field	Device Flag	indicates other information about the device such as multi-sensor, non-volatile memory control, protocol bridge, etc.
9	Unsigned24	Device ID	indicates a serial number for the device. The manufacturer shall assign a unique value for every device that has the identical values for Manufacturer ID and Expanded Device Type
12	Unsigned8	Preamble Count	indicates the minimum number of preambles to be sent with the response message from the slave to the master
13	Unsigned8	Variable Count	specifies the maximum number of objects (variables) that can be accessed from the device. The value of this parameter indicates the last variable code that a Client application can expect to be found in the device
14	Unsigned16	Configuration Change Count	keeps track of number of device configuration changes. The device shall increment the value of this parameter every time it receives a request to change the configuration using application layer services, or a user of the device changes the device configuration using local mean such as local operator's interface
16	Bit Field	Device ExtdStatus	indicates the extended operational status of the device
17	HCF enumeration	Manufacturer ID	is a 2 octet enumeration and indicates the manufacturer that produced the device. It shall be unique for Type 20 devices that are within one network and are in range of other networks
19	HCF enumeration	Distributor code	is a 2 octet enumeration and indicates the private label manufacturer that distributed the device. It shall be unique for Type 20 devices that are within one network and are in range of other networks
21	HCF enumeration	Device profile	specifies the class to which the device belongs

Table 8 – Identify command specific response codes

Value	Class	Description
0	Success	No error

5.3.2.3 Procedure at responding device

If the request contains a Tag or Long Tag, then the device has to compare it with the Tag or Long Tag stored in the device. The device shall respond only if the value in request matches the value stored in the device.

5.3.3 Read primary variable FAL PDU

5.3.3.1 Request primitive

The value of Command is one (1); the value of Octet Count field is zero (0) and Data field is null.

5.3.3.2 Response primitive

The Value field of response is shown in Table 9. The Response Code values are shown in Table 10.

Table 9 – Read primary variable response value field

Octet offset	Data type	Parameter name	Description
0	HCF enumeration	Primary Variable Unit	This parameter represents the engineering unit of primary variable.
1 – 4	Float32	Primary Variable	It is the current value of primary variable.

Table 10 – Read primary variable command specific response codes

Value	Class	Description
0	Success	No error
6	Error	Device-Specific Command Error
8	Warning	Update Failure
16	Error	Access Restricted

5.3.4 Read loop current and percent of range FAL PDU

5.3.4.1 Request primitive

The value of Command is two (2); the value of Octet Count field is zero (0) and Data field is null.

5.3.4.2 Response primitive

The Value field of response is shown in Table 11. The Response Code values are shown in Table 12.

Table 11 – Read loop current and percent of range value field

Octet offset	Data type	Parameter name	Description
0 – 3	Float32	Loop Current	It is the current value of the output of the device – either loop current in milli-amperes or voltage output in volts.
4 – 7	Float32	Primary Variable	It is the current value of primary variable in percent of range. It tracks the primary input to the device. Its value can be outside the normal range of 0% to 100%.

Table 12 – Read loop current and percent of range command specific response codes

Value	Class	Description
0	Success	No error
6	Error	Device-Specific Command Error
8	Warning	Update Failure
16	Error	Access Restricted

5.3.5 Read dynamic variables and loop current FAL PDU

5.3.5.1 Request primitive

The value of Command is three (3); the value of Octet Count field is zero (0) and Data field is null.

5.3.5.2 Response primitive

The Value field of response is shown in Table 13. The Response Code values are shown in Table 14. If a device does not support all four variables, then the response contains only the number of variables that the device supports. In that case, the remaining fields are absent from the response.

Table 13 – Read dynamic variables and loop current value field

Octet offset	Data type	Parameter name	Description
0 – 3	Float32	Loop Current	It is the current value of the output of the device – either loop current in milli-amperes or voltage output in volts.
4	HCF enumeration	Primary Variable Unit	This parameter represents the engineering unit of primary variable.
5 – 8	Float32	Primary Variable	It is the current value of primary variable.
9	HCF enumeration	Secondary Variable Unit	This parameter represents the engineering unit of secondary variable.
10 – 13	Float32	Secondary Variable	It is the current value of secondary variable.
14	HCF enumeration	Tertiary Variable Unit	This parameter represents the engineering unit of tertiary variable.
15 – 18	Float32	Tertiary Variable	It is the current value of tertiary variable.
19	HCF enumeration	Quaternary Variable Unit	This parameter represents the engineering unit of quaternary variable.
20 – 23	Float32	Quaternary Variable	It is the current value of quaternary variable.

Table 14 – Read dynamic variables and loop current command specific response codes

Value	Class	Description
0	Success	No error
6	Error	Device-Specific Command Error
8	Warning	Update Failure
16	Error	Access Restricted

5.3.6 Write loop configuration FAL PDU

5.3.6.1 Request primitive

The value of Command is six (6); the value of Octet Count field is two (2) and Data field is as shown in Table 15.

Table 15 – Write polling address value field

Octet offset	Data type	Parameter name	Description
0	Enumeration	Polling Address	This parameter represents the data-link address of the device.
1	Enumeration	Loop Current Mode	This parameter encodes the loop current mode as shown in Table 16.

Table 16 – Loop current mode codes

Value	Description
0	Disabled
1	Enabled
251	None
252	Unknown
253	Device specific

5.3.6.2 Response primitive

The Value field of response is shown in Table 15. The Response Code values are shown in Table 17.

Table 17 – Write polling address command specific response codes

Value	Class	Description
0	Success	No error
2	Error	Invalid Poll Address
5	Error	Too few data bytes received
6	Error	Device-Specific Command Error
7	Error	In write protect mode
12	Error	Invalid mode selection
16	Error	Access Restricted
32	Error	Busy

5.3.6.3 Procedure at responding device

A devices is required to disable loop current signalling when requested by Master. When current signalling is disabled, the loop current is set to the minimum value required for field device operation. The field device status bit 4 as shown in Table 3, Loop Current Fixed, is set to '1'.

5.3.7 Read loop configuration FAL PDU

5.3.7.1 Request primitive

The value of Command is seven (7); the value of Octet Count field is zero (0) and Data field is null.

5.3.7.2 Response primitive

The Value field of response is shown in Table 11. The Response Code values are shown in Table 12.

Table 18 – Read loop configuration value field

Octet offset	Data type	Parameter name	Description
0	Enumeration	Polling Address	This parameter represents the data-link address of the device.
1	Enumeration	Loop Current Mode	This parameter encodes the loop current mode as shown in Table 16.

Table 19 – Read loop configuration command specific response codes

Value	Class	Description
0	Success	No error
16	Error	Access Restricted

5.3.8 Read dynamic variable families classifications FAL PDU

5.3.8.1 Request primitive

The value of Command is eight (8); the value of Octet Count field is zero (0) and Data field is null.

5.3.8.2 Response primitive

The Value field of response is shown in Table 20. The Response Code values are shown in Table 21.

Table 20 – Read dynamic variable families classifications value field

Octet offset	Data type	Parameter name	Description
0	HCF enumeration	Primary Variable Classification	This parameter indicates the function performed by the device variable.
1	HCF enumeration	Secondary Variable Classification	This parameter indicates the function performed by the device variable.
2	HCF enumeration	Tertiary Variable Classification	This parameter indicates the function performed by the device variable.
3	HCF enumeration	Quaternary Variable Classification	This parameter indicates the function performed by the device variable.

Table 21 – Read dynamic variable families classifications command specific response codes

Value	Class	Description
0	Success	No error
16	Error	Access Restricted

5.3.9 Read device variables with status FAL PDU

5.3.9.1 Request primitive

The value of Command is nine (9); the value of Octet count field is one to eight (1 – 8) and it is equal to number of requested variables. The request Value field is shown in Table 22.

Table 22 – Read device variables with status request value field

Octet offset	Data type	Parameter name	Description
0	Unsigned8	Slot 0	This parameter is the code assigned to a device variable
1	Unsigned8	Slot 1	This parameter is the code assigned to a device variable
2	Unsigned8	Slot 2	This parameter is the code assigned to a device variable
3	Unsigned8	Slot 3	This parameter is the code assigned to a device variable
4	Unsigned8	Slot 4	This parameter is the code assigned to a device variable
5	Unsigned8	Slot 5	This parameter is the code assigned to a device variable
6	Unsigned8	Slot 6	This parameter is the code assigned to a device variable
7	Unsigned8	Slot 7	This parameter is the code assigned to a device variable

5.3.9.2 Response primitive

The Value field of response is shown in Table 23. The Response code values are shown in Table 25.

Table 23 – Read device variables with status value field

Octet offset	Data type	Parameter name	Description
0	Bit Field	Device ExtdStatus	This parameter indicates the extended operational status of the device
1	Unsigned8	Variable Code	This is the code received in request at Value field for slot 0
2	HCF enumeration	Variable classification	This indicates the function performed by the device variable for slot 0 as specified in IEC 62591(Ed.1.0), Table F.4
3	HCF enumeration	Variable Unit	This represents the engineering unit of variable for slot 0 as specified in ANNEX F TBD
4	Float32	Variable Value	It is the current value of variable for slot 0
8	Bit Field	Variable Status	This represents the status of variable for slot 0 as specified in Table 24
9	Unsigned8	Variable Code	This is the code received in request at Value field for slot 1
10	HCF enumeration	Variable classification	This indicates the function performed by the device variable for slot 1 as specified in ANNEX F TBD
11	HCF enumeration	Variable Unit	This represents the engineering unit of variable for slot 1 as specified in ANNEX F TBD
12	Float32	Variable Value	It is the current value of variable for slot 1
16	Bit Field	Variable Status	This represents the status of variable for slot 1 as specified in Table 24
17	Unsigned8	Variable Code	This is the code received in request at Value field for slot 2

Octet offset	Data type	Parameter name	Description
18	HCF enumeration	Variable classification	This indicates the function performed by the device variable for slot 2 as specified in ANNEX F TBD
19	HCF enumeration	Variable Unit	This represents the engineering unit of variable for slot 2 as specified in ANNEX F
20	Float32	Variable Value	It is the current value of variable for slot 2
24	Bit Field	Variable Status	This represents the status of variable for slot 2 as specified in Table 24
25	Unsigned8	Variable Code	This is the code received in request at Value field for slot 3
26	HCF enumeration	Variable classification	This indicates the function performed by the device variable for slot 3 as specified in ANNEX F TBD
27	HCF enumeration	Variable Unit	This represents the engineering unit of variable for slot 3 as specified in ANNEX F TBD
28	Float32	Variable Value	It is the current value of variable for slot 3
32	Bit Field	Variable Status	This represents the status of variable for slot 3 as specified in Table 24
33	Unsigned8	Variable Code	This is the code received in request at Value field for slot 4
34	HCF enumeration	Variable classification	This indicates the function performed by the device variable for slot 4 as specified in ANNEX F TBD
35	HCF enumeration	Variable Unit	This represents the engineering unit of variable for slot 4 as specified in ANNEX F TBD
36	Float32	Variable Value	It is the current value of variable for slot 4
40	Bit Field	Variable Status	This represents the status of variable for slot 4 as specified in Table 24
41	Unsigned8	Variable Code	This is the code received in request at Value field for slot 5
42	HCF enumeration	Variable classification	This indicates the function performed by the device variable for slot 5 as specified in ANNEX F TBD
43	HCF enumeration	Variable Unit	This represents the engineering unit of variable for slot 5 as specified in ANNEX F TBD
44	Float32	Variable Value	It is the current value of variable for slot 5
48	Bit Field	Variable Status	This represents the status of variable for slot 5 as specified in Table 24
49	Unsigned8	Variable Code	This is the code received in request at Value field for slot 6
50	HCF enumeration	Variable classification	This indicates the function performed by the device variable for slot 6 as specified in ANNEX F TBD
51	HCF enumeration	Variable Unit	This represents the engineering unit of variable for slot 6 as specified in ANNEX F TBD
52	Float32	Variable Value	It is the current value of variable for slot 6
56	Bit Field	Variable Status	This represents the status of variable for slot 6 as specified in Table 24
57	Unsigned8	Variable Code	This is the code received in request at Value field for slot 7
58	HCF enumeration	Variable classification	This indicates the function performed by the device variable for slot 7 as specified in ANNEX F TBD
59	HCF enumeration	Variable Unit	This represents the engineering unit of variable for slot 7 as specified in ANNEX F TBD
60	Float32	Variable Value	It is the current value of variable for slot 7
64	Bit Field	Variable Status	This represents the status of variable for slot 7 as specified in Table 24
65	Time	Time stamp	Slot 0 data time stamp

Variable Status field is Bit Field data type and it provides the quality status of the variable. Its encoding is shown in Table 24.

Table 24 – Variable status values

Bit mask	Definition
0xC0	These two bits are encoded to show the quality of the input process data: 11: good 01: poor accuracy 10: manual or fixed 00 bad
0x30	These two bits are encoded to show the limit status of the variable: 11: constant 01: low limited 10: high limited 00 not limited
0x08	If this bit is '1' then it indicates that more status values are available by reading device specific objects
0x07	The meaning of these bits is specific to the device family

Table 25 – Read device variables with status command specific response codes

Value	Class	Description
0	Success	No error
2	Error	Invalid selection
5	Error	Too few data octets received
6	Error	Device-Specific Command Error
8	Warning	Update Failure
14	Warning	Dynamic variables returned for device variables
16	Error	Access restricted
30	Warning	Command response truncated

5.3.9.3 Procedure at responding device

If the request does not contain any data octets (Value field) then the device shall respond with error code equal to 5; otherwise the respond code shall be a value other than 5.

The device shall respond with values of exactly as many variables as in the request.

If a variable requested is not supported in the device, then in the response APDU, the corresponding:

Variable classification field has to be set to '0';

Variable Unit field has to be set to '250';

Variable Value field has to be set to a floating point value of 'Not-a-number' (0x7F, 0xA0, 0x00, 0x00); and

Variable Status field has to be set to 0x30.

If the Variable classification is not supported for a requested variable, then Variable classification field in the response APDU for that variable has to be set to '0' (not yet classified) and the respond code shall be set to '14'.

The device shall support at least four slots.

5.3.10 Read message FAL PDU

5.3.10.1 Request primitive

The value of Command is twelve (12); the value of Octet Count field is zero (0) and Data field is null.

5.3.10.2 Response primitive

The Value field of response is shown in Table 26. The Response Code values are shown in Table 27.

Table 26 – Read message response value field

Octet offset	Data type	Parameter name	Description
0 – 23	Packed ASCII	Message	This is the value of Message stored in the device.

Table 27 – Read message command specific response codes

Value	Class	Description
0	Success	No error
16	Error	Access Restricted
32	Error	Busy

5.3.11 Read tag, descriptor and date FAL PDU

5.3.11.1 Request primitive

The value of Command is thirteen (13); the value of Octet Count field is zero (0) and Data field is null.

5.3.11.2 Response primitive

The Value field of response is shown in Table 28. The Response Code values are shown in Table 29.

Table 28 – Read tag, descriptor, date response value field

Octet offset	Data type	Parameter name	Description
0 – 5	Packed ASCII	Tag	This is the value of Tag stored in the device.
6 – 17	Packed ASCII	Descriptor	This is the value of Descriptor stored in the device.
18 – 20	Date	Date	This is the value of Date stored in the device.

Table 29 – Read tag, descriptor, date command specific response codes

Value	Class	Description
0	Success	No error
16	Error	Access Restricted
32	Error	Busy

5.3.12 Read primary variable transducer FAL PDU

5.3.12.1 Request primitive

The value of Command is fourteen (14); the value of Octet Count field is zero (0) and Data field is null.

5.3.12.2 Response primitive

The Value field of response is shown in Table 30. The Response Code values are shown in Table 31.

Table 30 – Read primary variable transducer information response value field

Octet offset	Data type	Parameter name	Description
0 – 2	Unsigned24	Transducer Serial Number	This is the value of Transducer Serial Number.
3	HCF enumeration	Transducer Unit	This is the code for Transducer Limits and Minimum Span Units
4 – 7	Float32	Upper Transducer Limit	It is the value of Upper Transducer Limit variable.
8 – 11	Float32	Lower Transducer Limit	It is the value of Lower Transducer Limit variable.
12 – 15	Float32	Minimum Span	It is the value of Minimum Span variable.

Table 31 – Read primary variable transducer information command specific response codes

Value	Class	Description
0	Success	No error
16	Error	Access Restricted
32	Error	Busy

5.3.12.3 Procedure at responding device

If the Transducer Serial Number is not applicable to the device, then the device shall respond with following values:

a. Transducer Serial Number equal to ‘0’;

b. Transducer Unit equal to ‘250’;

c. all other values equal to a floating point value of ‘Not-a-number’ (0x7F, 0xA0, 0x00, 0x00).

5.3.13 Read device information FAL PDU

5.3.13.1 Request primitive

The value of Command is fifteen (15); the value of Octet Count field is zero (0) and Data field is null.

5.3.13.2 Response primitive

The Value field of response is shown in Table 32. The Response Code values are shown in Table 33.

Table 32 – Read device information response value field

Octet offset	Data type	Parameter name	Description
0	HCF enumeration	PV Alarm Selection	This the value of PV Alarm Selection. It indicates the action taken by the device under error conditions. For transmitters, the code indicates the action taken by the Loop Current. For Actuators, the action taken by the positioner is indicated.
1	HCF enumeration	PV Transfer Function	This the value of PV Transfer Function. If the device does not support transfer functions, then this value is '0'.
2	HCF enumeration	PV Range Unit	This the value of PV upper and lower Range Unit.
3 – 6	Float32	PV Upper Range	PV Upper Range Value
7 – 10	Float32	PV Lower Range	PV Lower Range Value
11 – 14	Float32	PV Damping	PV Damping Value in seconds.
15	HCF enumeration	Write Protect	This the value of Write Protect. If the device does not implement write protection, then this value is '251'.
16	HCF enumeration	Private Label Distributor	This the value of Private Label Distributor. Its default value is the same as Manufacturer ID.
17	Bit Field	PV Analog Channel	This the value of PV Analog Channel flags.

Table 33 – Read device information command specific response codes

Value	Class	Description
0	Success	No error
16	Error	Access Restricted
32	Error	Busy

5.3.14 Read final assembly number FAL PDU

5.3.14.1 Request primitive

The value of Command is sixteen (16); the value of Octet Count field is zero (0) and Data field is null.

5.3.14.2 Response primitive

The Value field of response is shown in Table 34. The Response Code values are shown in Table 35.

Table 34 – Read final assembly number response value field

Octet offset	Data type	Parameter name	Description
0 – 2	Unsigned24	Final Assembly Number	This is the value of Final Assembly Number

Table 35 – Read final assembly number command specific response codes

Value	Class	Description
0	Success	No error
16	Error	Access Restricted
32	Error	Busy

5.3.15 Write message FAL PDU

5.3.15.1 Request primitive

The value of Command is seventeen (17); the value of Octet Count field is twenty-four (24) and Data field is as shown in Table 36.

Table 36 – Write message value field

Octet offset	Data type	Parameter name	Description
0 – 23	Packed ASCII	Message	This is a string used by a Master for record keeping

5.3.15.2 Response primitive

The Value field of response is shown in Table 36. The Response Code values are shown in Table 37.

Table 37 – Write message command specific response codes

Value	Class	Description
0	Success	No error
2	Error	Invalid Poll Address
5	Error	Too few data bytes received
6	Error	Device-Specific Command Error
7	Error	In write protect mode
16	Error	Access Restricted
32	Error	Busy

5.3.16 Write tag, descriptor and date FAL PDU

5.3.16.1 Request primitive

The value of Command is eighteen (18); the value of Octet Count field is twenty-one (21) and Data field is as shown in Table 38.

Table 38 – Write tag, descriptor, date value field

Octet offset	Data type	Parameter name	Description
0 – 5	Packed ASCII	Tag	This is the value of Tag to be stored in the device.
6 – 17	Packed ASCII	Descriptor	This is the value of Descriptor to be stored in the device.
18 – 20	Date	Date	This is the value of Date to be stored in the device.

5.3.16.2 Response primitive

The Value field of response is shown in Table 38. The Response Code values are shown in Table 39.

Table 39 – Write tag, descriptor, date command specific response codes

Value	Class	Description
0	Success	No error
5	Error	Too few data bytes received
6	Error	Device-Specific Command Error
7	Error	In write protect mode
9	Error	Invalid Date code
16	Error	Access Restricted
32	Error	Busy

5.3.17 Write final assembly number FAL PDU

5.3.17.1 Request primitive

The value of Command is nineteen (19); the value of Octet Count field is three (3) and Data field is as shown in Table 40.

Table 40 – Write final assembly number value field

Octet offset	Data type	Parameter name	Description
0 – 2	Unsigned24	Final Assembly Number	This is the value of Final Assembly Number. It is normally changed when electronics or other device components are upgraded in the field.

5.3.17.2 Response primitive

The Value field of response is shown in Table 40. The Response Code values are shown in Table 41.

Table 41 – Write final assembly number command specific response codes

Value	Class	Description
0	Success	No error
5	Error	Too few data bytes received
6	Error	Device-Specific Command Error
7	Error	In write protect mode
16	Error	Access Restricted
32	Error	Busy

5.3.18 Read long tag FAL PDU

5.3.18.1 Request primitive

The value of Command is twenty (20); the value of Octet Count field is zero (0) and Data field is null.

5.3.18.2 Response primitive

The Value field of response is shown in Table 42. The Response Code values are shown in Table 43.

Table 42 – Read long tag response value field

Octet offset	Data type	Parameter name	Description
0 – 31	Latin-1	Long Tag	This is the value of Long Tag stored in the device.

Table 43 – Read long tag command-specific response codes

Value	Class	Description
0	Success	No error
16	Error	Access Restricted
32	Error	Busy

5.3.19 Write long tag FAL PDU

5.3.19.1 Request primitive

The value of Command is twenty-two (22); the value of Octet Count field is thirty-two (32) and Data field is as shown in Table 44.

Table 44 – Write long tag value field

Octet offset	Data type	Parameter name	Description
0 – 31	Latin-1	Long Tag	This is the value of Long Tag to be stored in the device.

5.3.19.2 Response primitive

The Value field of response is shown in Table 44. The Response Code values are shown in Table 45.

Table 45 – Write long tag command specific Response codes

Value	Class	Description
0	Success	No error
5	Error	Too few data bytes received
6	Error	Device-Specific Command Error
7	Error	In write protect mode
16	Error	Access Restricted
32	Error	Busy

5.4 Data coding rules

5.4.1.1 General

If the user data is always a simple (primitive) component. It is encoded as shown in Figure 5. The semantics of the user data are known implicitly from the position in the PDU, the length is fixed and implicitly known.

Data

Figure 5 – Coding without identification

5.4.1.2 Coding for integer types

Integer values are signed quantities as shown in Figure 6 and Figure 7. The MSB of this data type is transferred first followed by the next octets and the LSB is transferred last.

Notation:	**Integer8, Integer16, Integer24, Integer32**		
Range of Values:	Data Type	range of values	length
	Integer8	$-128 \le i \le 127$	1 octet
	Integer16	$-32768 \le i \le 32767$	2 octets
	Integer24	$-2^{23} \le i \le 2^{23} - 1$	3 octets
	Integer32	$-2^{31} \le i \le 2^{31} - 1$	4 octets
Coding:	In two's complement representation the MSB (Most Significant Bit) is the bit after the sign bit (SN) in the first octet. SN = 0: positive numbers and z SN = 1: negative numbers		

Figure 6 – Coding of Integer type data

	bits	8	7	6	5	4	3	2	1
octets	1	SN	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
	2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Figure 7 – Coding of Integer16 type data

5.4.1.3 Coding for unsigned Types

Unsigned Values are encoded as shown in Figure 8 and Figure 9. The MSB of this data type is transferred first followed by the next octets and the LSB is transferred last.

Notation:	**Unsigned8, Unsigned16, Unsigned24, Unsigned32**		
Range of Values:	**Data Type**	**range of values**	**length**
	Unsigned8	$0 \le i \le 255$	1 octet
	Unsigned16	$0 \le i \le 65535$	2 octets
	Unsigned24	$0 \le i \le 2^{24} - 1$	3 octets
	Unsigned32	$0 \le i \le 2^{32} - 1$	4 octets
Coding	**Binary**		

Figure 8 – Coding of Unsigned type data

	bits	8	7	6	5	4	3	2	1
octets	1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
	2	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Figure 9 – Coding of Unsigned16 type data

5.4.1.4 Coding for floating point Type

Floating Point values are encoded as shown in Figure 10 and Figure 11. The sign and MSB of the exponent is transferred first followed by the balance of the exponent and the MSB to LSB of the fraction. If the value of an object of floating point data type is not known, then the value transferred is 0x7F, 0xA0, followed by all zeros (0x00); this value represents a 'Not-a-number'.

Notation:	**Floating-Point (4 octet)**
Range of Values:	see IEC 60559, Short Real Number (32 bits)
Coding:	see IEC 60559, Short Real Number (32 bits)

								LSB
bits	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
octets	**Exponent (E)**							
1	SN	2^7	2^6	2^5	2^4	2^3	2^2	2^1
	(E)	**Fraction (F)**						
2	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
	Fraction (F)							
3	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
	Fraction (F)							
4	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}

SN: sign 0 = positive, 1 = negative

Figure 10 – Coding of single precision Floating Point type data

Notation:	Floating-Point (8 octet)
Range of Values:	see IEC 60559, Short Real Number (64 bits)
Coding:	see IEC 60559, Short Real Number (64 bits)

LSB

bits	8	7	6	5	4	3	2	1
octets	Exponent (E)							
1	SN	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}
	Exponent (E)				Fraction (F)			
2	2^{3}	2^{2}	2^{1}	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	Fraction (F)							
3	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}
	Fraction (F)							
4	2^{-13}	2^{-14}	2^{-15}	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}
	Fraction (F)							
4	2^{-21}	2^{-22}	2^{-23}	2^{-24}	2^{-25}	2^{-26}	2^{-27}	2^{-28}
	Fraction (F)							
4	2^{-29}	2^{-30}	2^{-31}	2^{-32}	2^{-33}	2^{-34}	2^{-35}	2^{-36}
	Fraction (F)							
4	2^{-37}	2^{-38}	2^{-39}	2^{-40}	2^{-41}	2^{-42}	2^{-43}	2^{-44}
	Fraction (F)							
4	2^{-45}	2^{-46}	2^{-47}	2^{-48}	2^{-49}	2^{-50}	2^{-51}	2^{-52}

SN: sign 0 = positive, 1 = negative

Figure 11 – Coding of double precision Floating Point type data

5.4.1.5 Coding for date type

The data type Date consists of a calendar date as shown in Table 46 and Figure 12. The first octet representing day of month is transferred first, followed by the month octet and the year octet is transferred last.

Notation: Date
Range of Values: 1 January 1900 to 31 December 2155
Coding: in 3 octets

Table 46 – Coding for Date type

Parameter	Range of values	Meaning of the parameters
d. of m.	1...31	day of month
months	1...12	months
years	0...255	years – 1900

bits	8	7	6	5	4	3	2	1	
octets 1	RSV	RSV	RSV	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	1...31 d. of m.
2	RSV	RSV	RSV	RSV	2^{3}	2^{2}	2^{1}	2^{0}	1...12 months
3	RSV	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	0 ... 255 years

Figure 12 – Coding of Date type data

5.4.1.6 Coding for enumeration

The data type Enumeration is used to associate specific user definitions to numeric codes. The definitions are as shown in Table 47. The definitions for codes 250–255 is always as shown. It is not necessary to define all possible values in the table. All undefined codes in the table are reserved. Reserved values are not used, returned or transferred by any device. For every object of Enumeration type, there is such table that is known to the users by associating the table with a variable index.

Range of Values: 0 to 255
Coding: in 1 octet

Table 47 – Coding for one octet Enumerated Type

Unsigned value	Definition
0	Specified definition 0
1	Specified definition 1
⋮	⋮
n	Specified definition n
n+1	Reserved
n+2	Reserved
⋮	⋮
249	Reserved
250	"Not Used"
251	"None"
252	"Unknown"
253	"Special"
254	"Expansion"
255	Reserved

5.4.1.7 Coding for bit field

The Bit Field type of data is used to transfer objects encoded as single-bit data. Individual bits are packed together, and reserved bits are added as necessary to transfer one octet. A lookup table is used to define the meaning of each bit in the bit field as shown in Table 48. In other words, each bit shall be either true (set to a 1) or false (set to a 0). Any unused bit is reserved. For every object of Bit Field type, there is such table that is known to the users by associating the table with a variable index.

Range of Values: 0 to 255
Coding: in 1 octet

Table 48 – One octet bit field

Bit mask	Definition
0x01	For Bit 1 (LSB)
0x02	For Bit 2
0x04	For Bit 3
0x08	For Bit 4
0x10	For Bit 5
0x20	For Bit 6
0x40	For Bit 7
0x80	For Bit 8 (MSB)

5.4.1.8 Coding for packed ASCII string type

Packed ASCII is a 6-bit character code representing a subset of the ASCII character code set as shown in Table 49. It is produced by compressing four Packed ASCII characters into three octets. This type of string is always a multiple of 4 characters (3 octets) and is padded out to the end of the data item with SP characters. The length of the string is not transferred along with the string. The length is predetermined for each type of object, which is identified by object index. If the number of characters in the value of a string object is less than its predetermined length, then unused characters at the end of the string are set to SP (0x20). For example, 4 SP characters at the end of a string would appear as the 3 octets: 0x82, 0x08, and 0x20. The first character of this string is transferred first, followed by the next character and the last character is transferred last.

Table 49 – Packed ASCII character set

Bits 4 and 5	Bits 0 to 3															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
2	SP[a]	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?

[a] SP indicates a space character.

NOTE Most significant hexadecimal digit is top to bottom; least significant is left to right.

5.4.1.9 Coding for restricted ISO Latin-1 string type

ISO Latin-1 (ISO/IEC 8859-1) string consists of one character per octet. A restricted version that omits rows 0, 1, 8 and 9 in the Latin-1 table is used, as shown in Table 50.

The length of the string is not transferred along with the string. The length is predetermined for each type of object, which is identified by object index. If the number of characters in the value of a string object is less than its predetermined length, then unused characters at the end of the string are set to zeros (0x00). One zero (0x00) at the end of a string ISO Latin-1 data item may not be sufficient to meet this requirement. A valid restricted ISO Latin-1 character, other than zero, is permitted in the last octet of an restricted ISO Latin-1 data object. The first character of this string is transferred first, followed by the next character and the last character is transferred last.

Table 50 – AccepTable subset of ISO Latin-1 characters

Bits 4 to 7	Bits 0 to 3															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2	SP[a]	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	B	c	D	e	f	g	h	i	j	k	l	m	n	o
7	p	q	R	s	T	u	v	w	x	y	z	{	|	}	~	
8																
9																
A	NBSP[b]	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY[c]	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	Ä	å	æ	ç	è	é	Ê	ë	ì	ì	î	ï
F	ð	ñ	ò	ó	Ô	õ	ö	÷	ø	ù	Ú	û	ü	ý	þ	Ÿ

[a] SP indicates a space character.

[b] NBSP indicates a non-breaking space character.

[c] SHY indicates a soft hyphen.

NOTE Most significant hexadecimal digit is top to bottom, least significant is left to right. Grayed out cells means that no character is assigned to this code.

6 Structure of FAL protocol state machines

Interface to FAL services and protocol machines are specified in this subclause.

NOTE The state machines specified in this subclause and ARPMs defined in the following sections only define the valid events for each. It is a local matter to handle these invalid events.

The behaviour of the FAL is described by three integrated protocol machines. The three protocol machines are: FAL Service Protocol Machine (FSPM), the Application Relationship Protocol Machine (ARPM), and the data-link layer Mapping Protocol Machine (DMPM). The relationship among these protocol machines as well as primitives exchanged among them are depicted in depicted in Figure 13.

The FSPM describes the service interface between the FAL User and the AREP. The FSPM does not have any state changes. The FSPM is responsible for the following activities:

a) to accept service primitives from the FAL service user and convert them into FAL internal primitives;

b) to send FAL internal primitives to the ARPM;

c) to accept FAL internal primitives from the ARPM and convert them into service primitives for the FAL User;

d) to deliver the FAL service primitives to the FAL User.

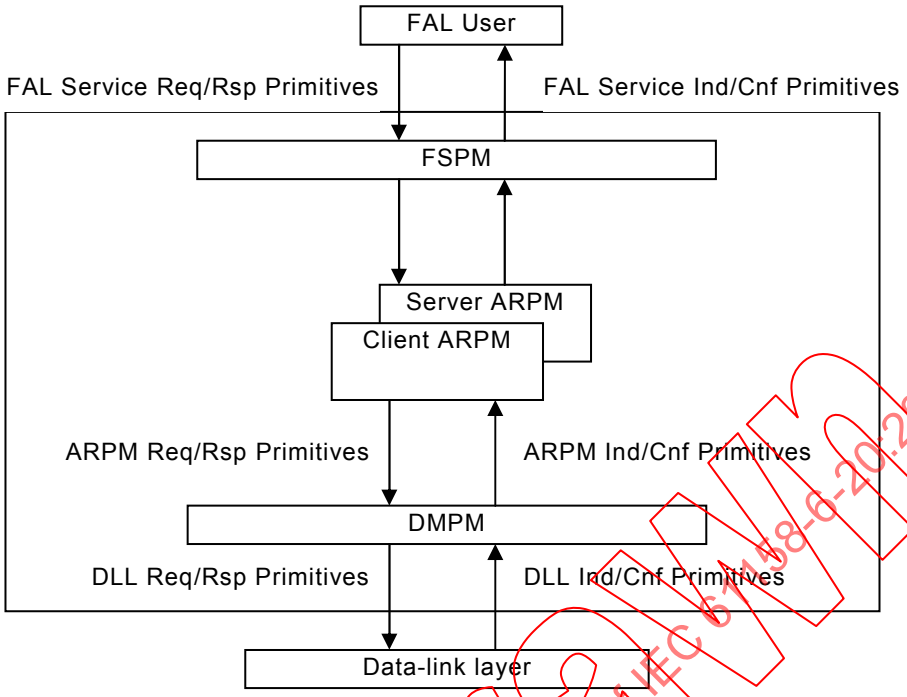


Figure 13 – Relationships among protocol machines and adjacent layers

The ARPM describes the establishment and release of an AR and exchange of FAL-PDUs with a remote ARPM(s).There are two types of ARPM – Client, Slave. In any FAL, there can be only one instance of ARPM. The type of ARPM depends upon the device class. The Master class device has Client ARPM; the Slave class device has Server ARPM. The ARPM is responsible for the following activities:

a) to accept FAL internal primitives from the FSPM and create and send other FAL internal primitives to either the FSPM or the DMPM, based on the primitive types;

b) to accept FAL internal primitives from the DMPM and send them to the FSPM as a form of FAL internal primitives;

The DMPM describes the mapping between the FAL and the DLL. It does not have any state changes. The DMPM is responsible for the following activities:

a) to accept FAL internal primitives from the ARPM, prepare DLL service primitives, and send them to the DLL;

b) to receive DLL indication or confirmation primitives from the DLL and send them to the ARPM in a form of FAL internal primitives.

7 AP-context state machines

There is no AP-Context State Machine in this Type of FAL.

8 FAL service protocol machine (FSPM)

8.1 General

FAL Service Protocol Machine is common for AL User services. It has one state called "ACTIVE" as shown in Figure 14.

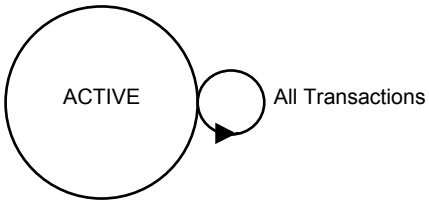


Figure 14 – State transition diagram of FSPM

8.2 FSPM state tables

Table 51 and Table 52 specify the FSPM protocol machine.

Table 51 – FSPM state Table – client transactions

#	Current state	Event or condition => action	Next state

S1	ACTIVE	Identify.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S2	ACTIVE	Read Primary Variable.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S3	ACTIVE	Read Loop Current.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S4	ACTIVE	Read Dynamic Variables.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S5	ACTIVE	Write Loop Configuration.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S6	ACTIVE	Read Loop Configuration.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S7	ACTIVE	Read Dynamic Variable Families Classifications.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S8	ACTIVE	Read Device Variables.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

#	Current state	**Event** or **condition** **=> action**	Next state

S9	ACTIVE	Read Message.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S10	ACTIVE	Read Tag, Descriptor, Date.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S11	ACTIVE	Read Primary Variable Transducer.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S12	ACTIVE	Read Device Information.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S13	ACTIVE	Read Final Assembly Number.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S14	ACTIVE	Write Message.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S15	ACTIVE	Write Tag, Descriptor, Date.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S16	ACTIVE	Write Final Assembly Number.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S17	ACTIVE	Read Long Tag.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S18	ACTIVE	Write Long Tag.req
		=>
		Transaction.req {
		remote_address := AREP_ID,
		user_data := alpdu
		}
			ACTIVE

S19	ACTIVE	Transaction.cnf
		&& CommErr (user_data) = "True"
		=>
		Identify.cnf(-) {
		Arep_id = remote_address,
		Comm_stat = Comm (user_data),
		Dev_stat = Device (user_data)
		}
			ACTIVE

#	Current state	Event or condition => action	Next state

S20	ACTIVE	Transaction.cnf
		&& CommandErr (user_data) = "True"
		=>
		Identify.cnf(-) {
		Arep_id = remote_address,
		Resp_code = Resp (user_data),
		Dev_stat = Device (user_data)
		}
			ACTIVE

S21	ACTIVE	Transaction.cnf
		&& CmdErr (user_data) = "False"
		&& Command (user_data) = 0 | 11 | 21
		=>
		Identify.cnf(+) {
		Arep_id = remote_address,
		Resp_code = Resp (user_data),
		Dev_stat = Device (user_data),
		data = Value (user_data)
		}
			ACTIVE

S22	ACTIVE	Transaction.cnf
		&& CmdErr (user_data) = "False"
		&& Command (user_data) = 1
		=>
		Read Primary Variable.cnf(+) {
		Arep_id = remote_address,
		Resp_code = Resp (user_data),
		Dev_stat = Device (user_data),
		data = Value (user_data)
		}
			ACTIVE

S23	ACTIVE	Transaction.cnf
		&& CmdErr (user_data) = "False"
		&& Command (user_data) = 2
		=>
		Read Loop Current.cnf(+) {
		Arep_id = remote_address,
		Resp_code = Resp (user_data),
		Dev_stat = Device (user_data),
		data = Value (user_data)
		}
			ACTIVE

S24	ACTIVE	Transaction.cnf
		&& CmdErr (user_data) = "False"
		&& Command (user_data) = 3
		=>
		Read Dynamic Variables.cnf(+) {
		Arep_id = remote_address,
		Resp_code = Resp (user_data),
		Dev_stat = Device (user_data),
		data = Value (user_data)
		}
			ACTIVE

S25	ACTIVE	Transaction.cnf
		&& CmdErr (user_data) = "False"
		&& Command (user_data) = 6
		=>
		Write Polling Address.cnf(+) {
		Arep_id = remote_address,
		Resp_code = Resp (user_data),
		Dev_stat = Device (user_data),
		data = Value (user_data)
		}
			ACTIVE

S26	ACTIVE	Transaction.cnf
		&& CmdErr (user_data) = "False"
		&& Command (user_data) = 7
		=>
		Read Loop Configuration.cnf(+) {
		Arep_id = remote_address,
		Resp_code = Resp (user_data),
		Dev_stat = Device (user_data),
		data = Value (user_data)
		}
			ACTIVE

#	Current state	**Event** or **condition** **=> action**	Next state

S27	ACTIVE	Transaction.cnf
		&& CmdErr (user_data) = "False"
		&& Command (user_data) = 8
		=>
		Read Dynamic Variable Families Classifications.cnf(+) {
		Arep_id = remote_address,
		Resp_code = Resp (user_data),
		Dev_stat = Device (user_data),
		data = Value (user_data)
		}
			ACTIVE

S28	ACTIVE	Transaction.cnf
		&& CmdErr (user_data) = "False"
		&& Command (user_data) = 9
		=>
		Read Device Variables.cnf(+) {
		Arep_id = remote_address,
		Resp_code = Resp (user_data),
		Dev_stat = Device (user_data),
		data = Value (user_data)
		}
			ACTIVE

S29	ACTIVE	Transaction.cnf
		&& CmdErr (user_data) = "False"
		&& Command (user_data) = 12
		=>
		Read Message.cnf(+) {
		Arep_id = remote_address,
		Resp_code = Resp (user_data),
		Dev_stat = Device (user_data),
		data = Value (user_data)
		}
			ACTIVE

S30	ACTIVE	Transaction.cnf
		&& CmdErr (user_data) = "False"
		&& Command (user_data) = 13
		=>
		Read Tag, Descriptor, Date.cnf(+) {
		Arep_id = remote_address,
		Resp_code = Resp (user_data),
		Dev_stat = Device (user_data),
		data = Value (user_data)
		}
			ACTIVE

S31	ACTIVE	Transaction.cnf
		&& CmdErr (user_data) = "False"
		&& Command (user_data) = 14
		=>
		Read Primary Variable Transducer.cnf(+) {
		Arep_id = remote_address,
		Resp_code = Resp (user_data),
		Dev_stat = Device (user_data),
		data = Value (user_data)
		}
			ACTIVE

S32	ACTIVE	Transaction.cnf
		&& CmdErr (user_data) = "False"
		&& Command (user_data) = 15
		=>
		Read Device Information.cnf(+) {
		Arep_id = remote_address,
		Resp_code = Resp (user_data),
		Dev_stat = Device (user_data),
		data = Value (user_data)
		}
			ACTIVE

S33	ACTIVE	Transaction.cnf
		&& CmdErr (user_data) = "False"
		&& Command (user_data) = 16
		=>
		Read Final Assembly Number.cnf(+) {
		Arep_id = remote_address,
		Resp_code = Resp (user_data),
		Dev_stat = Device (user_data),
		data = Value (user_data)
		}
			ACTIVE