DNP3 Application Note AN2013-004b
Validation of Incoming DNP3 Data

Revision History

Version	Date	Reason for Changes
AN2013-004	11-December-2013	Initial release.
AN2013-004a	29-June-2014	Correction to row 7 of Table 1: “is invalid or marked as “NOT_SUPPORTED”” was replaced with “is invalid or is not supported by the device”.
AN2013-004b	13-August-2014	Correction to row 17 of Table 1: Incoming broadcast messages shall be rejected if the control function code is anything other than 3 (CONFIRMED_USER_DATA) or 4 (UNCONFIRMED_USER_DATA). Removal of row 18 of Table 1 “The frame's DESTINATION field is set to one of the three broadcast addresses, and the FCV bit is set to 1.”.

1 Introduction

The stable and secure operation of SCADA systems is a high-priority issue. One component of SCADA robustness involves the validation of incoming messages. This Application Note details numerous checks that should be performed on received data, in order to verify that the data comprises a valid DNP3 message.

This document assumes that the reader is knowledgeable about the DNP3 Primary-Secondary transaction and the various components of a DNP3 message as specified in IEEE Std™ 1815-2012. Detailed explanations are included in IEEE 1815 and are not repeated in this document; references to specific sections in IEEE 1815 are included where appropriate. The figures included in this document are duplicated from Appendix B of IEEE 1815, and are reproduced here for ease of reference.

2 Details

This document refers to behavior of Master and Outstation devices. The validation that should be performed by a Master is not always the same as the validation that should be performed by an Outstation. Also, one device may perform the actions of both a Master and an Outstation (e.g. data concentrators). Depending on whether the incoming data was received by the Master or Outstation module, it may be necessary to validate the data differently.

Validation of incoming data is an essential element of good programming practices. This document includes general recommendations, as well as layer-dependent details for robust handling of data/message validation.

2.1 General Coding Recommendations

The recommendations below use the terms “robust” and “reasonable”. The implementer must determine what “robust” and “reasonable” mean in regards to the device being implemented.

Logging (what is logged, as well as changeable log levels) should be considered as a component of robust handling of error conditions.

Parse and validate all incoming data using a “trust nothing” approach before processing it or performing any actions based on the data.

The code must reflect the functionality of the device; the developer must know the device and what is to be implemented. Validation of object data should be robust and object-dependent; the handling of unexpected/invalid data will need to be device-dependent and well-thought-out (e.g. master requests that do not apply to an outstation’s specific functionality, timestamps that contain invalid values that cannot be converted to a realistic internal time, counter values that decrease, etc.).

Stop parsing when a non-recoverable failure (e.g. invalid function code) is encountered, and do not parse any further objects in the fragment.

Perform testing of boundary conditions, and combinations of boundary conditions. Perform a “reasonableness” test before allocating or consuming resources (e.g. unreasonable number of objects/points).

Verify that resource allocation has succeeded before attempting to use the resources. It is important to know how to check for the failure; it is not easy in all systems. Failure cases should be handled robustly.

Appropriate variable types should be used to avoid integer overflows, truncation, or negative numbers. Calculations based on index ranges require the use of integer types with a larger bit-width than the size of the index contained in the message (e.g. a count of objects in a 2-byte start/stop header could use a 32-bit integer for correct representation).

Avoid unintended sign extension by masking values when copying from smaller to larger integers. This especially applies when reading a value from a character array into an integer type. Unsigned declarations don’t always work as expected.

Ensure that sign bits and truncation are handled correctly when translating 48-bit DNP3 time to 32-bit internal timestamps.

Ensure that the parsing/processing of variable-length data, file transfer, etc. is robust in order to avoid accessing out-of-bound memory.

Consider the removal of debugging artifacts and associated constructs from the production code.

When discarding messages, all code artifacts should be ‘tidied up’ (e.g. buffers reset/cleared, variables reset to default values, etc.) so that subsequent processing does not encounter remnants from previous messages.

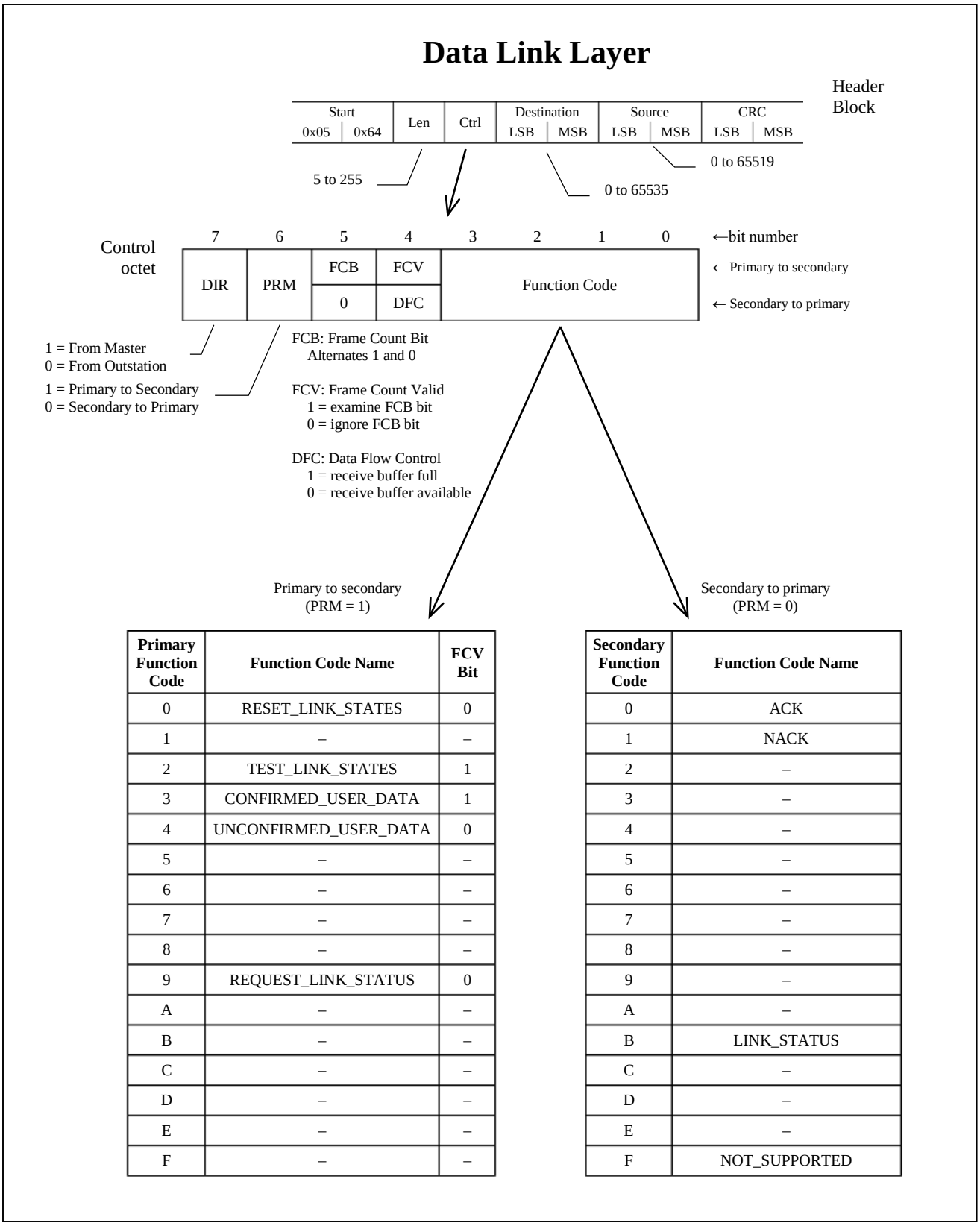
All fields of all data objects should be checked for "reasonable" or valid values (e.g. bits in flag words, values of code fields, etc.).

APIs should validate all data for applications. Types passed across API boundaries should be as strong as possible to prevent potential integration vulnerabilities.

Adopt a holistic Security Development Life-cycle (SDL) that includes, but is not limited to, code reviews, static & dynamic analysis, and fuzz testing (e.g. www.microsoft.com/security/sdl/, grouper.ieee.org/groups/plv/, www.cert.org/secure-coding/).

2.2 Data Link Layer

IEEE 1815 Clause 9 *Data Link Layer* provides a framework for implementing a Data Link Layer. Additional verifications and checks are required for a robust Data Link Layer.



Primary Function Code	Function Code Name	FCV Bit
0	RESET_LINK_STATES	0
1	–	–
2	TEST_LINK_STATES	1
3	CONFIRMED_USER_DATA	1
4	UNCONFIRMED_USER_DATA	0
5	–	–
6	–	–
7	–	–
8	–	–
9	REQUEST_LINK_STATUS	0
A	–	–
B	–	–
C	–	–
D	–	–
E	–	–
F	–	–

Secondary Function Code	Function Code Name
0	ACK
1	NACK
2	–
3	–
4	–
5	–
6	–
7	–
8	–
9	–
A	–
B	LINK_STATUS
C	–
D	–
E	–
F	NOT_SUPPORTED

Figure 1 - Data link layer header breakout

Table 1 details conditions that shall cause incoming data to be ignored (discarded) by the Data Link Layer. After data has been discarded, the monitoring of the incoming data to locate a new start-of-message (0x05 0x64) shall recommence with the data received immediately after the discarded data.

The Master & Outstation columns in Table 1 serve several purposes. An "N/A" entry in a cell indicates that the feature does not apply to a device of that type. Developers may wish to enter a checkmark in each blank cell as they verify and test code.

Table 1 Erroneous data which shall cause incoming data to be ignored by the Data Link Layer

Ignore (discard) the received data if:	Master	Outstation	IEEE 1815-2012 Reference
The first octet is not set to 0x05.			9.2.4.1.1
The second octet is not set to 0x64.			9.2.4.1.1
The value of the third octet (the frame's LENGTH field) is less than 5.			9.2.4.1.2
The value of the third octet (the frame's LENGTH field) plus 3 plus the count of CRC octets is greater than the device's maximum received frame size.			9.2.4, 9.2.4.1.2
The value of the frame's LENGTH field plus 3 plus the count of CRC octets is greater than the actual number of octets received.			9.2.4, 9.2.4.1.2
Any of the 16-bit CRC fields is incorrect (1 to 16 CRC fields per frame, depending on the frame size).			9.2.4, 9.2.4.3, Annex E
The combination of PRM bit, FCV bit and function code in the CONTROL field (fourth octet of the frame) is invalid or is not supported by the device.			9.2.4.1.3; Annex B *Valid Data Link Layer Control Codes*
The frame's CONTROL function code is set to 3 or 4 (USER_DATA), and the value of the LENGTH field is less than 6.			
The value of the frame's CONTROL function code is something other than 3 or 4, and the value of the LENGTH field is not equal to 5.			
The frame's PRM bit is clear (Secondary message), but the device is not currently expecting a Secondary message.			9.2.4.1.3.2
The Secondary message contains a CONTROL function code that is not a permitted response to the function code that was used in the Primary message.			Table 9-1
The FCV bit in the CONTROL field is set to 1, and the FCB bit does not match the expected value. The previous Data Link response must be retransmitted.			9.2.4.1.3.3, 9.2.4.1.3.4
The frame's DESTINATION field is set to a value between 0xFFF0 and 0xFFFB inclusive.			Technical Bulletin 2013-002
The frame's DESTINATION field is set to 0xFFFC and the self-address feature is currently disabled.			Technical Bulletin 2013-002
The frame's DESTINATION field is set to a value other than the Master's DNP3 address or 0xFFFC.		N/A	Technical Bulletin 2013-002
The frame's DESTINATION field is set to a value other than the Outstation's DNP3 address, one of the three broadcast addresses (0xFFFF, 0xFFFE, 0xFFFD), or 0xFFFC.	N/A		Technical Bulletin 2013-002
The frame's DESTINATION field is set to one of the three broadcast addresses, and the CONTROL function code is set to anything other than 3 (CONFIRMED_USER_DATA) or 4 (UNCONFIRMED_USER_DATA) with PRM set.	N/A		9.2.4.1.3.6, 9.2.4.1.4, Technical Bulletin 2013-002
The frame's SOURCE field is set to a value between 0xFFF0 and 0xFFFF inclusive.			9.2.4.1.5, 9.2.5.2.3

Robust device-dependent solutions must also be developed to handle possible complications involving the Data Link Layer that can occur in the field:

- Accept RESET_LINK_STATES between frames of multi-segment messages.
- Accept REQUEST_LINK_STATUS between frames of multi-segment messages.
- Accept TEST_LINK_STATES between frames of multi-segment messages.
- Accept application layer data sent in a mixture of confirmed and unconfirmed data link frames (frame's CONTROL function code set to 3 or 4).
- Accept or discard a frame which contains gaps between octets in a serial transmission.
- Accept or discard a frame which contains a CONTROL function code set to 3 or 4 (USER_DATA) and the value of the LENGTH field is less than 7.
- Accept or discard a frame which contains SOURCE and DESTINATION fields set to the same value.
- Accept or discard messages from unknown sources (SOURCE field).

2.3 Transport Function

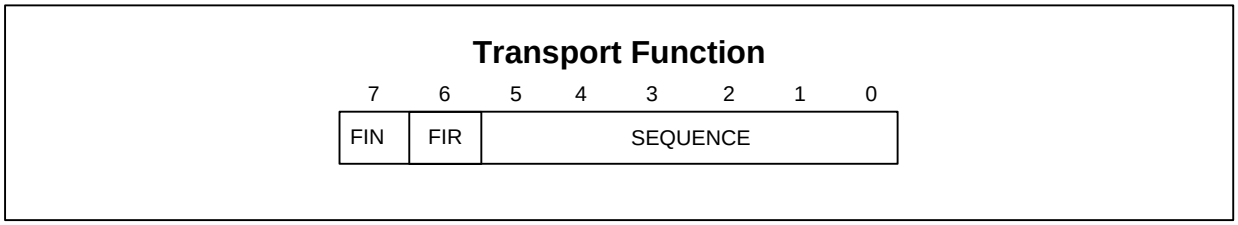
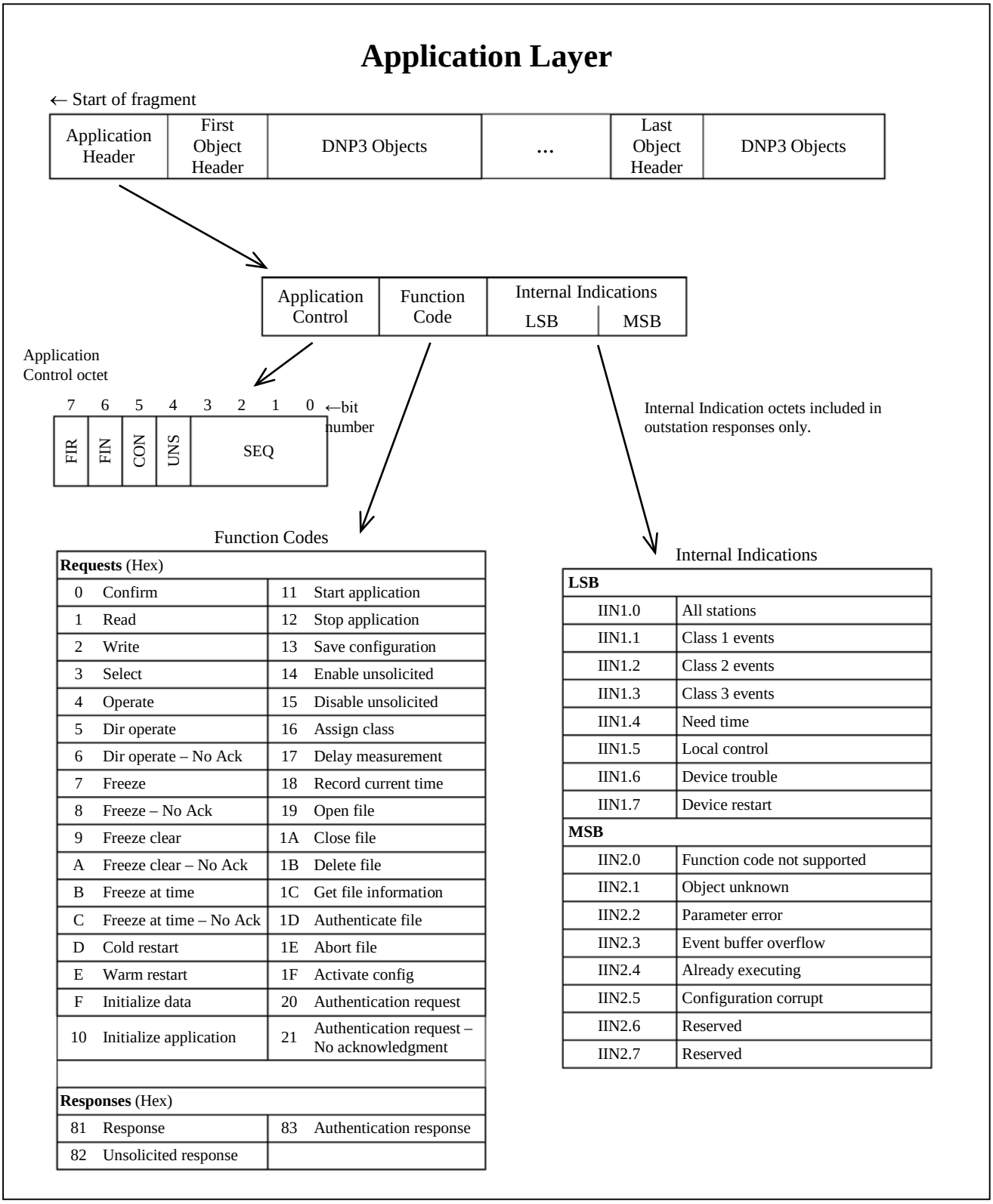


Figure 2 - Transport function header breakout

IEEE 1815-2012 Clause 8 *Transport Function*, and Technical Bulletin 2013-003 *Updated Transport Reception State Table*, together provide the information required to implement a robust Transport Function.

2.4 Application Layer

IEEE 1815-2012 Clause 6 *Application Layer* contains the state tables and diagrams that provide a framework for the Application Layer. Additional verifications and checks are required for a robust Application Layer.



Requests (Hex)			
0	Confirm	11	Start application
1	Read	12	Stop application
2	Write	13	Save configuration
3	Select	14	Enable unsolicited
4	Operate	15	Disable unsolicited
5	Dir operate	16	Assign class
6	Dir operate – No Ack	17	Delay measurement
7	Freeze	18	Record current time
8	Freeze – No Ack	19	Open file
9	Freeze clear	1A	Close file
A	Freeze clear – No Ack	1B	Delete file
B	Freeze at time	1C	Get file information
C	Freeze at time – No Ack	1D	Authenticate file
D	Cold restart	1E	Abort file
E	Warm restart	1F	Activate config
F	Initialize data	20	Authentication request
10	Initialize application	21	Authentication request – No acknowledgment
Responses (Hex)			
81	Response	83	Authentication response
82	Unsolicited response		

LSB	
IIN1.0	All stations
IIN1.1	Class 1 events
IIN1.2	Class 2 events
IIN1.3	Class 3 events
IIN1.4	Need time
IIN1.5	Local control
IIN1.6	Device trouble
IIN1.7	Device restart
MSB	
IIN2.0	Function code not supported
IIN2.1	Object unknown
IIN2.2	Parameter error
IIN2.3	Event buffer overflow
IIN2.4	Already executing
IIN2.5	Configuration corrupt
IIN2.6	Reserved
IIN2.7	Reserved

Figure 3 - Application layer header breakout

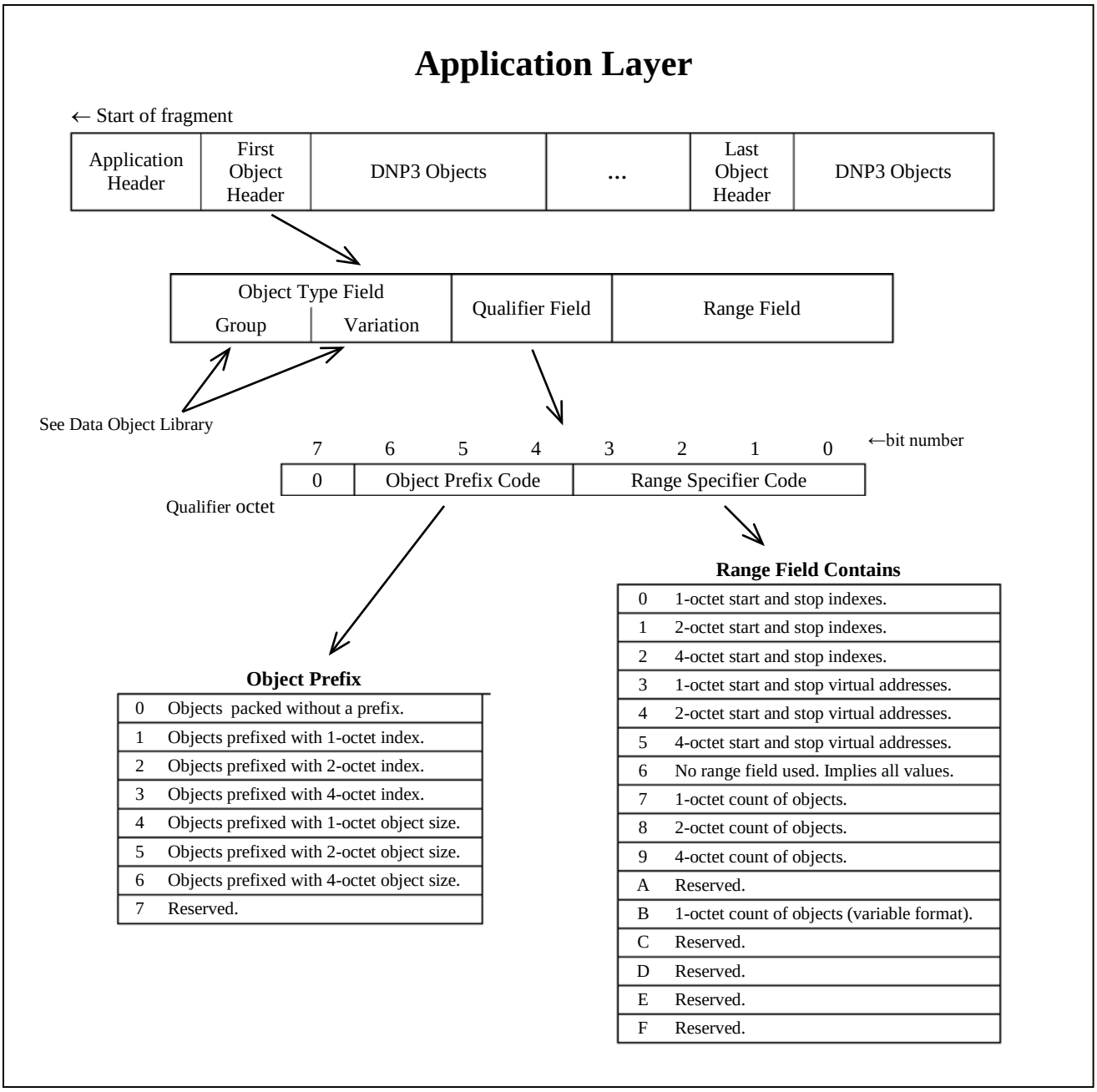


Figure 4 - Object header breakout

Each Application Layer fragment shall begin with one complete application header. Therefore, any fragment received by a Master that contains fewer than 4 octets shall be discarded, and any fragment received by an Outstation that contains fewer than 2 octets shall be discarded. These size checks must be performed before performing any other verifications of a received fragment.

The Master and Outstation state tables contained in IEEE 1815-2012 Clause 6 specify the behavior of a Master and Outstation when fragments are received, with the appropriate behavior determined primarily by reception timeouts and the contents of the application control field. The Outstation Fragment State Table also includes rudimentary verification of the Application Layer function code.

After the appropriate state table verifications have been completed, the objects contained in the fragment, in combination with the function code, must be parsed. A framework to verify and process the objects in a

fragment is detailed in the Master and Outstation fragment parsing pseudo-code below. This pseudo-code is not necessarily exhaustive.

2.4.1 Parsing a fragment received by a Master

Parsing begins with the first octet following the fragment's application header.

Variables

DataSize .....Total size of object data.
FC .....Function code, as specified in the application layer header.
FIN .....FINAL field, as specified in the application control octet of the application header.
FIR .....FIRST field, as specified in the application control octet of the application header.
GRP .....Object group, as specified in the 1st octet of the object header.
INDX .....Index (point number).
IndexSize .....Size of each object's index.
NumberObjects ...Number of objects, as specified in the range field of the object header.
ObjectSize.....Size of each object.
NOTE: Some objects are 1 or 2 bits in size, as opposed to the majority of objects which are a minimum of 1 octet in size.
OPC .....Object prefix code, as specified in bits 4-6 of QC.
RangeSize .....Number of octets in the range field, as determined from the RSC.
RSC .....Range specifier code, as specified in bits 0-3 of QC.
STRT.....Start point number, as specified in the range field.
STOP .....Stop point number, as specified in the range field.
QC .....Qualifier code, as specified in the 3rd octet of object header.
UnparsedSize.....Number of octets remaining to be processed;
Initially set to (fragment size – application header size).
VAR.....Object variation, as specified in the 2nd octet of the object header.

Pseudo-Code

InitialVerification

Consider the originating request from the Master: Is the response FC invalid?
    Discard the entire fragment, and stop parsing (do not transmit an application confirmation).

UnparsedSize = 0?
    Device does not support FC?
        Discard the entire fragment, and stop parsing (do not transmit an application confirmation).
    FC requires Objects?
        Discard the entire fragment, and stop parsing (do not transmit an application confirmation).
    FC does not require Objects (see Table 2)?
        Perform the appropriate action(s) indicated by the IIN bits and the function code, transmit an application confirmation if requested, and stop parsing.

ApplicationDataIteration

UnparsedSize < 3?
    Discard the remainder of the fragment buffer, and stop parsing (do not transmit an application confirmation).

Determine GRP from the 1st octet of the object header.
Determine VAR from the 2nd octet of the object header.
Determine QC from the 3rd octet of the object header.
Determine RSC from the 3rd octet of the object header.
Determine RangeSize from the RSC.

Device does not support GRP & VAR?
    Discard the remainder of the fragment buffer, and stop parsing (do not transmit an application confirmation).

Device does not support FC in combination with GRP & VAR?
    Discard the remainder of the fragment buffer, and stop parsing (do not transmit an application confirmation).

Consider the originating request from the Master: Is the combination of response FC, GRP & VAR invalid?
    Discard the remainder of the fragment buffer, and stop parsing (do not transmit an application confirmation).

Unsolicited message: Is the combination of response FC, GRP & VAR invalid?
    Discard the remainder of the fragment buffer, and stop parsing (do not transmit an application confirmation).

Device does not support FC for the combination of GRP, VAR, & QC?
Discard the remainder of the fragment buffer, and stop parsing (do not transmit an application confirmation).

Calculate UnparsedSize = UnparsedSize – 3.
UnparsedSize < RangeSize?
Discard the remainder of the fragment buffer, and stop parsing (do not transmit an application confirmation).

Calculate UnparsedSize = UnparsedSize – RangeSize.

RSC = 0, 1, 2, 3, 4, or 5?
Retrieve STRT and STOP from the range field.

STRT > STOP?
Discard the remainder of the fragment buffer, and stop parsing (do not transmit an application confirmation).

Calculate NumberObjects = STOP – STRT + 1.
NOTE: Appropriate variable types must be used to ensure that integer overflows and similar errors do not occur during this calculation (e.g. 65535 – 0 + 1).

Determine ObjectSize from the combination of GRP and VAR (see Table 3).

Calculate DataSize = NumberObjects * ObjectSize.
NOTE: If ObjectSize is measured in bits, DataSize must be converted to octets:
DataSize = DataSize / 8; add 1 if the modulo is non-zero.

UnparsedSize < DataSize?
Discard the remainder of the fragment buffer, and stop parsing (do not transmit an application confirmation).

For each object (as indicated by NumberObjects):
Parse the object data, validating individual fields as appropriate to verify that the values are valid, etc.

Calculate UnparsedSize = UnparsedSize – DataSize.

or RSC = 7, 8, or 9?
Retrieve NumberObjects from the range field.
Determine ObjectSize from the combination of GRP and VAR (see Table 3).
Determine IndexSize from the RSC.

Calculate DataSize = NumberObjects * (ObjectSize + IndexSize).
NOTE: If ObjectSize is measured in bits, DataSize must be converted to octets:
DataSize = DataSize / 8; add 1 if the modulo is non-zero.

UnparsedSize < DataSize?
Discard the remainder of the fragment buffer, and stop parsing (do not transmit an application confirmation).

For each object (as indicated by NumberObjects):
Parse the object data, validating individual fields as appropriate to verify that the values are valid, etc.

Calculate UnparsedSize = UnparsedSize – DataSize.

or RSC = 0xB (11)?
Retrieve NumberObjects from the range field.
Determine ObjectSize from the OPC.

For each object (as indicated by NumberObjects):
UnparsedSize < ObjectSize?
Discard the remainder of the fragment buffer, and stop parsing (do not transmit an application confirmation).

Calculate UnparsedSize = UnparsedSize – ObjectSize.
Retrieve DataSize (as specified by ObjectSize) from the unparsed object data.

UnparsedSize < DataSize?
Discard the remainder of the fragment buffer, and stop parsing (do not transmit an application confirmation).

Parse the object data, validating individual fields as appropriate to verify that the values are valid, etc.
Calculate UnparsedSize = UnparsedSize – DataSize.

UnparsedSize = 0?
Perform the appropriate action(s) indicated by the IIN bits, and the combination of function code and application data.Transmit an application confirmation if requested, and stop parsing.

Parse the remainder of the fragment: Return to **ApplicationDataIteration**.

2.4.2 Parsing a fragment received by an Outstation

Parsing begins with the first octet following the fragment's application header.

Variables

DataSize .....Total size of object data.
FC .....Function code, as specified in the application layer header.
FIN .....FINAL field, as specified in the application control octet of the application header.
FIR .....FIRST field, as specified in the application control octet of the application header.
GRP .....Object group, as specified in the 1st octet of the object header.
INDX .....Index (point number).
IndexSize .....Size of each object's index.
NumberObjects ...Number of objects, as specified in the range field of the object header.
ObjectSize.....Size of each object.
NOTE: Some objects are 1 or 2 bits in size, as opposed to the majority of objects which are a minimum of 1 octet in size.
OPC .....Object prefix code, as specified in bits 4-6 of QC.
RangeSize .....Number of octets in the range field, as determined from the RSC.
RSC .....Range specifier code, as specified in bits 0-3 of QC.
STRT.....Start point number, as specified in the range field.
STOP .....Stop point number, as specified in the range field.
QC .....Qualifier code, as specified in the 3rd octet of object header.
UnparsedSize.....Number of octets remaining to be processed;
Initially set to (fragment size – application header size).
VAR.....Object variation, as specified in the 2nd octet of the object header.

Pseudo-Code

InitialVerification

UnparsedSize = 0?
  Device does not support FC?
    Discard the entire fragment, set IIN2.0, transmit a null error message[a], and stop parsing.
  FC requires Objects?
    Discard the entire fragment, set IIN2.2, transmit a null error message[a], and stop parsing.
  FC does not require Objects (see Table 2)?
    Perform the appropriate action(s), transmit an appropriate response message[a], and stop parsing.

ApplicationDataIteration

UnparsedSize < 3?
  Discard the remainder of the fragment buffer, set IIN2.2, transmit a null error message[a], and stop parsing.

Determine GRP from the 1st octet of the object header.
Determine VAR from the 2nd octet of the object header.
Determine QC from the 3rd octet of the object header.
Determine RSC from the 3rd octet of the object header.
Determine RangeSize from the RSC.

Device does not support FC in combination with GRP & VAR?
  Depending on the FC:
    Either continue parsing (e.g. READ request for an event object supported by the outstation's subset level),
    Or discard the remainder of the fragment buffer, set IIN2.1, transmit a null error message[a], and stop parsing.

Device does not support FC for the combination of GRP, VAR, & QC?
  Discard the remainder of the fragment buffer, set IIN2.2, transmit a null error message[a], and stop parsing.

Calculate UnparsedSize = UnparsedSize – 3.

UnparsedSize < RangeSize?
  Discard the remainder of the fragment buffer, set IIN2.2, transmit a null error message[a], and stop parsing.

RSC = 6?
  No object data to parse for this object.

or RSC = 0, 1, 2, 3, 4, or 5?
  Retrieve STRT and STOP from the range field.
  STRT > STOP?

Discard the remainder of the fragment buffer, set IIN2.2, transmit a null error message[a], and stop parsing.

STRT or STOP outside device's point range for the specified GRP & VAR?
Set IIN2.2 in the response message.
Devices are encouraged to perform robust handling, perhaps by substituting the subset STRT and STOP and continuing to parse, but it is acceptable to discard the remainder of the fragment buffer, transmit a null error message[a], and stop parsing.

Calculate NumberObjects = STOP – STRT + 1.
NOTE: Appropriate variable types must be used to ensure that integer overflows and similar errors do not occur during this calculation (e.g. 65535 – 0 + 1).

Determine ObjectSize from the combination of GRP and VAR (see Table 3).
Calculate DataSize = NumberObjects * ObjectSize.
NOTE: If ObjectSize is measured in bits, DataSize must be converted to octets:
DataSize = DataSize / 8; add 1 if the modulo is non-zero.

UnparsedSize < DataSize?
Discard the remainder of the fragment buffer, set IIN2.2, transmit a null error message[a], and stop parsing.

For each object (as indicated by NumberObjects):
Parse the object data, validating individual fields as appropriate to verify that the values are valid, etc.

Calculate UnparsedSize = UnparsedSize – DataSize.

or RSC = 7, 8, or 9?
Retrieve NumberObjects from the range field.
Determine IndexSize from the RSC.
Calculate UnparsedSize = UnparsedSize – RangeSize.

Calculate DataSize = NumberObjects * (ObjectSize + IndexSize).
NOTE: If ObjectSize is measured in bits, DataSize must be converted to octets:
DataSize = DataSize / 8; add 1 if the modulo is non-zero.

UnparsedSize < DataSize?
Discard the remainder of the fragment buffer, set IIN2.2, transmit a null error message[a], and stop parsing.

For each object (as indicated by NumberObjects):
Parse the object data, validating individual fields as appropriate to verify that the values are valid, etc.

Calculate UnparsedSize = UnparsedSize – DataSize.

or RSC = 0xB (11)?
Retrieve NumberObjects from the range field.
Determine ObjectSize from the OPC.

For each object (as indicated by NumberObjects):
UnparsedSize < ObjectSize?
Discard the remainder of the fragment buffer, set IIN2.2, transmit a null error message[a], and stop parsing.

Calculate UnparsedSize = UnparsedSize – ObjectSize.
Retrieve DataSize (as specified by ObjectSize) from the unparsed object data.

UnparsedSize < DataSize?
Discard the remainder of the fragment buffer, set IIN2.2, transmit a null error message[a], and stop parsing.

Parse the object data, validating individual fields as appropriate to verify that the values are valid, etc.
Calculate UnparsedSize = UnparsedSize – DataSize.

UnparsedSize = 0?
Perform the appropriate action(s), transmit an appropriate response message[a], and stop parsing.

Parse the remainder of the fragment: Return to **ApplicationDataIteration**.

Notes
[a] Outstations shall never respond to a broadcast message or a NO-ACK request (e.g. DIRECT_OPERATE_NR), even in the case of errors.

2.4.3 Supplementary Tables

Table 2 lists the type of message in which an Application Function code is valid. It also specifies whether an application function code can "stand alone" in a valid message, or whether a supplementary object header is required.

Table 2 Function code summary

FC	Message Type	Name	Requires objects?
0 (0x00)	Confirmation	CONFIRM	Never
1 (0x01)	Request	READ	Always
2 (0x02)	Request	WRITE	Always
3 (0x03)	Request	SELECT	Always
4 (0x04)	Request	OPERATE	Always
5 (0x05)	Request	DIRECT_OPERATE	Always
6 (0x06)	Request	DIRECT_OPERATE_NR	Always
7 (0x07)	Request	IMMED_FREEZE	Always
8 (0x08)	Request	IMMED_FREEZE_NR	Always
9 (0x09)	Request	FREEZE_CLEAR	Always
10 (0x0A)	Request	FREEZE_CLEAR_NR	Always
11 (0x0B)	Request	FREEZE_AT_TIME	Always
12 (0x0C)	Request	FREEZE_AT_TIME_NR	Always
13 (0x0D)	Request	COLD_RESTART	Never
14 (0x0E)	Request	WARM_RESTART	Never
15 (0x0F)	Request (Obsolete)	INITIALIZE_DATA	Always
16 (0x10)	Request	INITIALIZE_APPL	Always
17 (0x11)	Request	START_APPL	Always
18 (0x12)	Request	STOP_APPL	Always
19 (0x13)	Request (Deprecated)	SAVE_CONFIG	Never
20 (0x14)	Request	ENABLE_UNSOLICITED	Always
21 (0x15)	Request	DISABLE_UNSOLICITED	Always
22 (0x16)	Request	ASSIGN_CLASS	Always
23 (0x17)	Request	DELAY_MEASURE	Never
24 (0x18)	Request	RECORD_CURRENT_TIME	Never
25 (0x19)	Request	OPEN_FILE	Always
26 (0x1A)	Request	CLOSE_FILE	Always
27 (0x1B)	Request	DELETE_FILE	Always
28 (0x1C)	Request	GET_FILE_INFO	Always
29 (0x1D)	Request	AUTHENTICATE_FILE	Always
30 (0x1E)	Request	ABORT_FILE	Always
31 (0x1F)	Request	ACTIVATE_CONFIG	Always
32 (0x20)	Request	AUTHENTICATE_REQ	Always
33 (0x21)	Request	AUTH_REQ_NO_ACK	Always
129 (0x81)	Response	RESPONSE	Usually
130 (0x82)	Response	UNSOLICITED_RESPONSE	Usually
131 (0x83)	Response	AUTHENTICATE_RESP	Always

Table 3 details each valid DNP3 object, and which function codes can be used with the object in both a request and response. It also lists the size of each object, which should be used to verify the size of a message.

Table 3 Object definition summary

GRP	VAR	Type	Description	Size	Request FC	Response FC
0 (0x00)	**196** (0xC4)	Attribute	Device Attributes - Configuration ID		1	129
0 (0x00)	**197** (0xC5)	Attribute	Device Attributes - Configuration version		1	129
0 (0x00)	**198** (0xC6)	Attribute	Device Attributes - Configuration build date		1	129
0 (0x00)	**199** (0xC7)	Attribute	Device Attributes - Configuration last change date		1	129
0 (0x00)	**200** (0xC8)	Attribute	Device Attributes - Configuration signature		1	129
0 (0x00)	**201** (0xC9)	Attribute	Device Attributes - Configuration signature algorithm		1	129
0 (0x00)	**202** (0xCA)	Attribute	Device Attributes - Master Resource ID (mRID)		1	129
0 (0x00)	**203** (0xCB)	Attribute	Device Attributes - Device altitude		1	129
0 (0x00)	**204** (0xCC)	Attribute	Device Attributes - Device longitude		1	129
0 (0x00)	**205** (0xCD)	Attribute	Device Attributes - Device latitude		1	129
0 (0x00)	**206** (0xCE)	Attribute	Device Attributes - User-assigned secondary operator name		1	129
0 (0x00)	**207** (0xCF)	Attribute	Device Attributes - User-assigned primary operator name		1	129
0 (0x00)	**208** (0xD0)	Attribute	Device Attributes - User-assigned system name		1	129
0 (0x00)	**209** (0xD1)	Attribute	Device Attributes - Secure authentication version		1	129
0 (0x00)	**210** (0xD2)	Attribute	Device Attributes - Number of security statistics per association		1	129
0 (0x00)	**211** (0xD3)	Attribute	Device Attributes - Identifier of support for user-specific Attributes		1	129
0 (0x00)	**212** (0xD4)	Attribute	Device Attributes - Number of master-defined data set prototypes		1	129
0 (0x00)	**213** (0xD5)	Attribute	Device Attributes - Number of outstation-defined data set prototypes		1	129
0 (0x00)	**214** (0xD6)	Attribute	Device Attributes - Number of master-defined data sets		1	129
0 (0x00)	**215** (0xD7)	Attribute	Device Attributes - Number of outstation-defined data sets		1	129
0 (0x00)	**216** (0xD8)	Attribute	Device Attributes - Max number of binary outputs per request		1	129
0 (0x00)	**217** (0xD9)	Attribute	Device Attributes - Local timing accuracy		1	129
0 (0x00)	**218** (0xDA)	Attribute	Device Attributes - Duration of timing accuracy		1	129

GRP	VAR	Type	Description	Size	Request FC	Response FC
0 (0x00)	**219** (0xDB)	Attribute	Device Attributes - Support for analog output events		1	129
0 (0x00)	**220** (0xDC)	Attribute	Device Attributes - Max analog output index		1	129
0 (0x00)	**221** (0xDD)	Attribute	Device Attributes - Number of analog outputs		1	129
0 (0x00)	**222** (0xDE)	Attribute	Device Attributes - Support for binary output events		1	129
0 (0x00)	**223** (0xDF)	Attribute	Device Attributes - Max binary output index		1	129
0 (0x00)	**224** (0xE0)	Attribute	Device Attributes - Number of binary outputs		1	129
0 (0x00)	**225** (0xE1)	Attribute	Device Attributes - Support for frozen counter events		1	129
0 (0x00)	**226** (0xE2)	Attribute	Device Attributes - Support for frozen counters		1	129
0 (0x00)	**227** (0xE3)	Attribute	Device Attributes - Support for counter events		1	129
0 (0x00)	**228** (0xE4)	Attribute	Device Attributes - Max counter index		1	129
0 (0x00)	**229** (0xE5)	Attribute	Device Attributes - Number of counter points		1	129
0 (0x00)	**230** (0xE6)	Attribute	Device Attributes - Support for frozen analog inputs		1	129
0 (0x00)	**231** (0xE7)	Attribute	Device Attributes - Support for analog input events		1	129
0 (0x00)	**232** (0xE8)	Attribute	Device Attributes - Maximum analog input index		1	129
0 (0x00)	**233** (0xE9)	Attribute	Device Attributes - Number of analog input points		1	129
0 (0x00)	**234** (0xEA)	Attribute	Device Attributes - Support for double-bit binary input events		1	129
0 (0x00)	**235** (0xEB)	Attribute	Device Attributes - Maximum double-bit binary input index		1	129
0 (0x00)	**236** (0xEC)	Attribute	Device Attributes - Number of double-bit binary input points		1	129
0 (0x00)	**237** (0xED)	Attribute	Device Attributes - Support for binary input events		1	129
0 (0x00)	**238** (0xEE)	Attribute	Device Attributes - Max binary input index		1	129
0 (0x00)	**239** (0xEF)	Attribute	Device Attributes - Number of binary input points		1	129
0 (0x00)	**240** (0xF0)	Attribute	Device Attributes - Max transmit fragment size		1, 2	129
0 (0x00)	**241** (0xF1)	Attribute	Device Attributes - Max receive fragment size		1	129
0 (0x00)	**242** (0xF2)	Attribute	Device Attributes - Device manufacturer’s software version		1	129
0 (0x00)	**243** (0xF3)	Attribute	Device Attributes - Device manufacturer’s hardware version		1	129
0 (0x00)	**244** (0xF4)	Attribute	Device Attributes - User-assigned owner name		1	129
0 (0x00)	**245** (0xF5)	Attribute	Device Attributes - User-assigned location name		1, 2	129

GRP	VAR	Type	Description	Size	Request FC	Response FC
0 (0x00)	**246** (0xF6)	Attribute	Device Attributes - User-assigned ID code/number		1, 2	129
0 (0x00)	**247** (0xF7)	Attribute	Device Attributes - User-assigned device name		1, 2	129
0 (0x00)	**248** (0xF8)	Attribute	Device Attributes - Device serial number		1	129
0 (0x00)	**249** (0XF9)	Attribute	Device Attributes - DNP subset and conformance		1	129
0 (0x00)	**250** (0xFA)	Attribute	Device Attributes - Device manufacturer's product name and model		1	129
0 (0x00)	**252** (0xFC)	Attribute	Device Attributes - Device manufacturer's name		1	129
0 (0x00)	**254** (0xFE)	Attribute	Device Attributes - Non-specific all Attributes request		1	129
0 (0x00)	**255** (0xFF)	Attribute	Device Attributes - List of Attribute variations		1	129
1 (0x01)	**0** (0x00)	Static	Binary Input - Any Variations		1, 22	
1 (0x01)	**1** (0x01)	Static	Binary Input - Packed Format	1 bit	1	129
1 (0x01)	**2** (0x02)	Static	Binary Input - Status with Flags	1 octet	1	129
2 (0x02)	**0** (0x00)	Event	Binary Input Event - Any Variations		1	
2 (0x02)	**1** (0x01)	Event	Binary Input Event	1 octet	1	129,130
2 (0x02)	**2** (0x02)	Event	Binary Input Event - with Absolute Time	7 octets	1	129,130
2 (0x02)	**3** (0x03)	Event	Binary Input Event - with Relative Time	3 octets	1	129,130
3 (0x03)	**0** (0x00)	Static	Double-bit Binary Input - Any Variations		1, 22	
3 (0x03)	**1** (0x01)	Static	Double-bit Binary Input - Packed Format	2 bits	1	129
3 (0x03)	**2** (0x02)	Static	Double-bit Binary Input - Status with Flags	1 octet	1	129
4 (0x04)	**0** (0x00)	Event	Double-bit Binary Input Event - Any Variations		1	
4 (0x04)	**1** (0x01)	Event	Double-bit Binary Input Event	1 octet	1	129,130
4 (0x04)	**2** (0x02)	Event	Double-bit Binary Input Event with Absolute Time	7 octets	1	129,130
4 (0x04)	**3** (0x03)	Event	Double-bit Binary Input Event with Relative Time	3 octets	1	129,130
10 (0x0A)	**0** (0x00)	Static	Binary Output - Any Variations		1, 22	
10 (0x0A)	**1** (0x01)	Static	Binary Output - Packed Format	1 bit	1, 2	129
10 (0x0A)	**2** (0x02)	Static	Binary Output - Status with Flags	1 octet	1	129
11 (0x0B)	**0** (0x00)	Event	Binary Output Event - Any Variations		1	
11 (0x0B)	**1** (0x01)	Event	Binary Output Event - Status	1 octet	1	129,130

GRP	VAR	Type	Description	Size	Request FC	Response FC
11 (0x0B)	**2** (0x02)	Event	Binary Output Event - Status with Time	7 octets	1	129,130
12 (0x0C)	**0** (0x00)	Command	Binary Output Command - Any Variations		22	
12 (0x0C)	**1** (0x01)	Command	Binary Output Command - Control Relay Output Block	11 octets	3, 4, 5, 6	129
12 (0x0C)	**2** (0x02)	Command	Binary Output Command - Pattern Control Block	11 octets	3, 4, 5, 6	129
12 (0x0C)	**3** (0X03)	Command	Binary Output Command - Pattern Mask	n bits	3, 4, 5, 6	129
13 (0x0D)	**0** (0x00)	Event	Binary Output Command Event - Any Variations		1	
13 (0x0D)	**1** (0x01)	Event	Binary Output Command Event - Command Status	1 octet	1	129,130
13 (0x0D)	**2** (0x02)	Event	Binary Output Command Event - Command Status with Time	7 octets	1	129,130
20 (0x14)	**0** (0x00)	Static	Counter - Any Variations		1,7,8,9,10,11,12,22	
20 (0x14)	**1** (0x01)	Static	Counter - 32-bit with Flag	5 octets	1	129
20 (0x14)	**2** (0x02)	Static	Counter - 16-bit with Flag	3 octets	1	129
20 (0x14)	**5** (0x05)	Static	Counter - 32-bit w/o Flag	4 octets	1	129
20 (0x14)	**6** (0x06)	Static	Counter - 16-bit w/o Flag	2 octets	1	129
21 (0x15)	**0** (0x00)	Static	Frozen Counter - Any Variations		1, 22	
21 (0x15)	**1** (0x01)	Static	Frozen Counter - 32-bit with Flag	5 octets	1	129
21 (0x15)	**2** (0x02)	Static	Frozen Counter - 16-bit with Flag	3 octets	1	129
21 (0x15)	**5** (0x05)	Static	Frozen Counter - 32-bit with Flag and Time	11 octets	1	129
21 (0x15)	**6** (0x06)	Static	Frozen Counter - 16-bit with Flag and Time	9 octets	1	129
21 (0x15)	**9** (0x09)	Static	Frozen Counter - 32-bit w/o Flag	4 octets	1	129
21 (0x15)	**10** (0x0A)	Static	Frozen Counter - 16-bit w/o Flag	2 octets	1	129
22 (0x16)	**0** (0x00)	Event	Counter Event - Any Variations		1	
22 (0x16)	**1** (0x01)	Event	Counter Event - 32-bit with Flag	5 octets	1	129, 130
22 (0x16)	**2** (0x02)	Event	Counter Event - 16-bit with Flag	3 octets	1	129, 130
22 (0x16)	**5** (0x05)	Event	Counter Event - 32-bit with Flag and Time	11 octets	1	129, 130
22 (0x16)	**6** (0x06)	Event	Counter Event - 16-bit with Flag and Time	9 octets	1	129, 130
23 (0x17)	**0** (0x00)	Event	Frozen Counter Event - Any Variations		1	
23 (0x17)	**1** (0x01)	Event	Frozen Counter Event - 32-bit with Flag	5 octets	1	129, 130

GRP	VAR	Type	Description	Size	Request FC	Response FC
23 (0x17)	**2** (0x02)	Event	Frozen Counter Event - 16-bit with Flag	3 octets	1	129, 130
23 (0x17)	**5** (0x05)	Event	Frozen Counter Event - 32-bit with Flag and Time	11 octets	1	129, 130
23 (0x17)	**6** (0x06)	Event	Frozen Counter Event - 16-bit with Flag and Time	9 octets	1	129, 130
30 (0x1E)	**0** (0x00)	Static	Analog Input - Any Variations		1, 7, 8, 11, 22	
30 (0x1E)	**1** (0x01)	Static	Analog Input - 32-bit with Flag	5 octets	1	129
30 (0x1E)	**2** (0x02)	Static	Analog Input - 16-bit with Flag	3 octets	1	129
30 (0x1E)	**3** (0x03)	Static	Analog Input - 32-bit w/o Flag	4 octets	1	129
30 (0x1E)	**4** (0x04)	Static	Analog Input - 16-bit w/o Flag	2 octets	1	129
30 (0x1E)	**5** (0x05)	Static	Analog Input - Single-prec. FP with Flag	5 octets	1	129
30 (0x1E)	**6** (0x06)	Static	Analog Input - Double-prec. FP with Flag	9 octets	1	129
31 (0x1F)	**0** (0x00)	Static	Frozen Analog Input - Any Variations		1, 22	
31 (0x1F)	**1** (0x01)	Static	Frozen Analog Input - 32-bit with Flag	5 octets	1	129
31 (0x1F)	**2** (0x02)	Static	Frozen Analog Input - 16-bit with Flag	3 octets	1	129
31 (0x1F)	**3** (0x03)	Static	Frozen Analog Input - 32-bit with Time-of-Freeze	11 octets	1	129
31 (0x1F)	**4** (0x04)	Static	Frozen Analog Input - 16-bit with Time-of-Freeze	9 octets	1	129
31 (0x1F)	**5** (0x05)	Static	Frozen Analog Input - 32-bit w/o Flag	4 octets	1	129
31 (0x1F)	**6** (0x06)	Static	Frozen Analog Input - 16-bit w/o Flag	2 octets	1	129
31 (0x1F)	**7** (0x07)	Static	Frozen Analog Input - Single-prec. FP with Flag	5 octets	1	129
31 (0x1F)	**8** (0x08)	Static	Frozen Analog Input - Double-prec. FP with Flag	9 octets	1	129
32 (0x20)	**0** (0x00)	Event	Analog Input Event - Any Variations		1	
32 (0x20)	**1** (0x01)	Event	Analog Input Event - 32-bit	5 octets	1	129, 130
32 (0x20)	**2** (0x02)	Event	Analog Input Event - 16-bit	3 octets	1	129,130
32 (0x20)	**3** (0x03)	Event	Analog Input Event - 32-bit with Time	11 octets	1	129,130
32 (0x20)	**4** (0x04)	Event	Analog Input Event - 16-bit with Time	9 octets	1	129,130
32 (0x20)	**5** (0x05)	Event	Analog Input Event - Single-prec. FP	5 octets	1	129,130
32 (0x20)	**6** (0x06)	Event	Analog Input Event - Double-prec. FP	9 octets	1	129,130
32 (0x20)	**7** (0x07)	Event	Analog Input Event - Single-prec. FP with Time	11 octets	1	129, 130

GRP	VAR	Type	Description	Size	Request FC	Response FC
32 (0x20)	**8** (0x08)	Event	Analog Input Event - Double-prec. FP with Time	15 octets	1	129, 130
33 (0x21)	**0** (0x00)	Event	Frozen Analog Input Event - Any Variations		1	
33 (0x21)	**1** (0x01)	Event	Frozen Analog Input Event - 32-bit	5 octets	1	129, 130
33 (0x21)	**2** (0x02)	Event	Frozen Analog Input Event - 16-bit	3 octets	1	129,130
33 (0x21)	**3** (0x03)	Event	Frozen Analog Input Event - 32-bit with Time	11 octets	1	129,130
33 (0x21)	**4** (0x04)	Event	Frozen Analog Input Event - 16-bit with Time	9 octets	1	129,130
33 (0x21)	**5** (0x05)	Event	Frozen Analog Input Event - Single-prec. FP	5 octets	1	129,130
33 (0x21)	**6** (0x06)	Event	Frozen Analog Input Event - Double-prec. FP	9 octets	1	129,130
33 (0x21)	**7** (0x07)	Event	Frozen Analog Input Event - Single-prec. FP with Time	11 octets	1	129, 130
33 (0x21)	**8** (0x08)	Event	Frozen Analog Input Event - Double-prec. FP with Time	15 octets	1	129, 130
34 (0x22)	**0** (0x00)	Static	Analog Input Deadband - Any Variations		1	
34 (0x22)	**1** (0x01)	Static	Analog Input Deadband - 16-bit	2 octets	1, 2	129
34 (0x22)	**2** (0x02)	Static	Analog Input Deadband - 32-bit	4 octets	1, 2	129
34 (0x22)	**3** (0x03)	Static	Analog Input Deadband - Single-prec. FP	4 octets	1, 2	129
40 (0x28)	**0** (0x00)	Static	Analog Output Status - Any Variations		1, 22	
40 (0x28)	**1** (0x01)	Static	Analog Output Status - 32-bit with Flag	5 octets	1	129
40 (0x28)	**2** (0x02)	Static	Analog Output Status - 16-bit with Flag	3 octets	1	129
40 (0x28)	**3** (0x03)	Static	Analog Output Status - Single-prec. FP with Flag	5 octets	1	129
40 (0x28)	**4** (0x04)	Static	Analog Output Status - Double-prec. FP with Flag	9 octets	1	129
41 (0x29)	**0** (0x00)	Command	Analog Output Command - Any Variations		22	
41 (0x29)	**1** (0x01)	Command	Analog Output Command - 32-bit	5 octets	3, 4, 5, 6	129
41 (0x29)	**2** (0x02)	Command	Analog Output Command - 16-bit	3 octets	3, 4, 5, 6	129
41 (0x29)	**3** (0x03)	Command	Analog Output Command - Single-prec. FP	5 octets	3, 4, 5, 6	129
41 (0x29)	**4** (0x04)	Command	Analog Output Command - Double-prec. FP	9 octets	3, 4, 5, 6	129
42 (0x2A)	**0** (0x00)	Event	Analog Output Event - Any Variations		1	
42 (0x2A)	**1** (0x01)	Event	Analog Output Event - 32-bit	5 octets	1	129, 130
42 (0x2A)	**2** (0x02)	Event	Analog Output Event - 16-bit	3 octets	1	129, 130

GRP	VAR	Type	Description	Size	Request FC	Response FC
42 (0x2A)	**3** (0x03)	Event	Analog Output Event - 32-bit with Time	11 octets	1	129, 130
42 (0x2A)	**4** (0x04)	Event	Analog Output Event - 16-bit with Time	9 octets	1	129, 130
42 (0x2A)	**5** (0x05)	Event	Analog Output Event - Single-prec. FP	5 octets	1	129, 130
42 (0x2A)	**6** (0x06)	Event	Analog Output Event - Double-prec. FP	9 octets	1	129, 130
42 (0x2A)	**7** (0x07)	Event	Analog Output Event - Single-prec. FP with Time	11 octets	1	129, 130
42 (0x2A)	**8** (0x08)	Event	Analog Output Event - Double-prec. FP with Time	15 octets	1	129, 130
43 (0x2B)	**0** (0x00)	Event	Analog Output Command Event - Any Variations		1	
43 (0x2B)	**1** (0x01)	Event	Analog Output Command Event - 32-bit	5 octets	1	129, 130
43 (0x2B)	**2** (0x02)	Event	Analog Output Command Event - 16-bit	3 octets	1	129, 130
43 (0x2B)	**3** (0x03)	Event	Analog Output Command Event - 32-bit with Time	11 octets	1	129, 130
43 (0x2B)	**4** (0x04)	Event	Analog Output Command Event - 16-bit with Time	9 octets	1	129, 130
43 (0x2B)	**5** (0x05)	Event	Analog Output Command Event - Single-prec. FP	5 octets	1	129, 130
43 (0x2B)	**6** (0x06)	Event	Analog Output Command Event - Double-prec. FP	9 octets	1	129, 130
43 (0x2B)	**7** (0x07)	Event	Analog Output Command Event - Single-prec. FP with Time	11 octets	1	129, 130
43 (0x2B)	**8** (0x08)	Event	Analog Output Command Event - Double-prec. FP with Time	15 octets	1	129, 130
50 (0x32)	**1** (0x01)	Info	Time and Date - Absolute Time	6 octets	1, 2	129
50 (0x32)	**2** (0x02)	Info	Time and Date - Absolute Time and Interval	10 octets	11, 12	
50 (0x32)	**3** (0x03)	Info	Time and Date - Absolute Time at Last Recorded Time	6 octets	2	
50 (0x32)	**4** (0x04)	Info	Time and Date - Indexed Absolute Time and Long Interval	11 octets	1, 2	129
51 (0x33)	**1** (0x01)	Info	Time and Date CTO - Absolute Time, Synchronized	6 octets		129, 130
51 (0x33)	**2** (0x02)	Info	Time and Date CTO - Absolute Time, Unsynchronized	6 octets		129, 130
52 (0x34)	**1** (0x01)	Info	Time Delay Coarse	2 octets		129
52 (0x34)	**2** (0x02)	Info	Time Delay Fine	2 octets		129
60 (0x3C)	**1** (0x01)	Info	Class Objects - Class 0 Data		1, 22	
60 (0x3C)	**2** (0x02)	Info	Class Objects - Class 1 Data		1, 20, 21, 22	
60 (0x3C)	**3** (0x03)	Info	Class Objects - Class 2 Data		1, 20, 21, 22	
60 (0x3C)	**4** (0x04)	Info	Class Objects - Class 3 Data		1, 20, 21, 22	

GRP	VAR	Type	Description	Size	Request FC	Response FC
70 (0x46)	**2** (0x02)	Info	File-Control - Authentication		29	129
70 (0x46)	**3** (0x03)	Info	File-Control - File Command		25, 26, 27	
70 (0x46)	**4** (0x04)	Info	File-Control - File Command Status		30	129, 130
70 (0x46)	**5** (0x05)	Info	File-Control - File Transport		1, 2	129, 130
70 (0x46)	**6** (0x06)	Info	File-Control - File Transport Status		1	129, 130
70 (0x46)	**7** (0x07)	Info	File-Control - File Descriptor		28	129, 130
70 (0x46)	**8** (0x08)	Info	File-Control - File Specification String		31	
80 (0x50)	**1** (0x01)	Static	Internal Indications - Packed Format	2 octets	1, 2	129
81 (0x51)	**1** (0x01)	Info	Device Storage - Buffer Fill Status	3 octets	1	129
82 (0x52)	**1** (0x01)	Info	Device Profile - Functions and Indexes			129, 130
83 (0x53)	**1** (0x01)	Any	Data Set - Private Registration Object		1	129, 130
83 (0x53)	**2** (0x02)	Info	Data Set - Private Registration Object Descriptor		1	129
85 (0x55)	**0** (0x00)	Info	Data Set Prototype - Any Variations		1, 2	
85 (0x55)	**1** (0x01)	Info	Data Set Prototype - With UUID		1	129, 130
86 (0x56)	**0** (0x00)	Info	Data Set Descriptor - Any Variations		22	129
86 (0x56)	**1** (0x01)	Info	Data Set Descriptor - Data Set Contents		1, 2	129
86 (0x56)	**2** (0x02)	Info	Data Set Descriptor - Characteristics	1 octet	1	129
86 (0x56)	**3** (0x03)	Info	Data Set Descriptor - Point Index Attributes		1, 2	129
87 (0x57)	**1** (0x01)	Static	Data Set - Any Variations		1	129
87 (0x57)	**1** (0x01)	Static	Data Set - Present Value		1, 2, 3, 4, 5, 6	129
88 (0x58)	**0** (0x00)	Event	Data Set Event - Any Variations		1	
88 (0x58)	**1** (0x01)	Event	Data Set Event - Snapshot		1	129, 130
90 (0x5A)	**1** (0x01)	Info	Application - Identifier		16, 17, 18	
91 (0x5B)	**1** (0x01)	Info	Status of Requested Operation			129
101 (0x65)	**1** (0x01)	Static	Binary-Coded Decimal Integer - Small	2 octets	1, 2	129
101 (0x65)	**2** (0x02)	Static	Binary-Coded Decimal Integer - Medium	4 octets	1, 2	129
101 (0x65)	**3** (0x03)	Static	Binary-Coded Decimal Integer - Large	8 octets	1, 2	129

GRP	VAR	Type	Description	Size	Request FC	Response FC
102 (0x66)	**1** (0x01)	Static	Unsigned Integer - 8-bit	1 octet	1, 2	129
110 (0x6E)	ALL	Static	Octet String		1, 2, 22	129
111 (0x6F)	ALL	Event	Octet String		1	129,130
112 (0x70)	ALL	Static	Virtual Terminal Output Block		2	
113 (0x71)	**0** (0x00)	Event	Virtual Terminal Event Data - Any Variations		1, 22	
113 (0x71)	ALL	Event	Virtual Terminal Event Data			129, 130
120 (0x78)	**0** (0x00)	Info	Authentication		22	
120 (0x78)	**1** (0x01)	Info	Authentication - Challenge		32	131
120 (0x78)	**2** (0x02)	Info	Authentication - Reply		32	131
120 (0x78)	**3** (0x03)	Info	Authentication - Aggressive Mode Request		1-31 (Any)	129,130
120 (0x78)	**4** (0x04)	Info	Authentication - Session Key Status Request	1 octet	32	
120 (0x78)	**5** (0x05)	Info	Authentication - Session key Status			131
120 (0x78)	**6** (0x06)	Info	Authentication - Session Key Change		32	
120 (0x78)	**7** (0x07)	Info/Event	Authentication - Error		33	131
120 (0x78)	**8** (0x08)	Info	Authentication - User Certificate		32	
120 (0x78)	**9** (0x09)	Info	Authentication - Message Authentication Code (MAC)		1-31 (Any)	129,130
120 (0x78)	**10** (0x0A)	Info	Authentication - User Status Change		32	
120 (0x78)	**11** (0x0B)	Info	Authentication - Update Key Change Request		32	
120 (0x78)	**12** (0x0C)	Info	Authentication - Update Key Change Reply			131
120 (0x78)	**13** (0x0D)	Info	Authentication - Update Key Change		32	
120 (0x78)	**14** (0x0E)	Info	Authentication - Update Key Change Signature		32	
120 (0x78)	**15** (0x0F)	Info	Authentication - Update Key Change Confirmation		32	131
121 (0x79)	**0** (0x00)	Info	Security Statistic		1, 22	
121 (0x79)	**1** (0x01)	Static	Security Statistic - 32-bit with Flag	7 octets	1	129
122 (0x7A)	**0** (0x00)	Info	Security Statistic Event		1	
122 (0x7A)	**1** (0x01)	Event	Security Statistic Event - 32-bit with Flag	7 octets	1	129,130
122 (0x7A)	**2** (0x02)	Event	Security Statistic Event - 32-bit with Flag and Time	13 octets	1	129,130

3 Conclusions

Thorough validation of incoming DNP3 data will contribute to the robust behavior of DNP3 devices, which will in turn enhance the security of SCADA systems. Vendors of DNP3 devices can use the checks detailed in this document as a starting point to ensure the robust implementation of DNP3 in their devices. Vendors should also use the IED Conformance Tests as a tool to verify that their DNP3 outstation devices will be interoperable in the field. The IED Conformance Tests can be downloaded from dnp.org.

4 Submitted by

The DNP Technical Committee.

5 Disclaimer

Application Notes contain application information developed by users and are provided for the benefit of other users. This note illustrates how the features of DNP3 are used to meet specific user requirements. This Application Note has been reviewed by the DNP Technical Committee. It does not contain all of the details that are mandatory for a complete DNP3 implementation, and the Committee does not warrant that the approach taken is the only way to use DNP3 to meet the user requirements.