IEEE Standard for Electric Power Systems Communications—Distributed Network Protocol (DNP3)

IEEE Power & Energy Society

Sponsored by the
Transmission and Distribution Committee

1815™

IEEE
3 Park Avenue
New York, NY 10016-5997, USA

1 July 2010

IEEE Std 1815™-2010

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

IEEE Standard for Electric Power Systems Communications—Distributed Network Protocol (DNP3)

Sponsor

Transmission and Distribution (PE/T&D) Committee
Substations Technical Committee

of the

IEEE Power & Energy Society

Approved 17 June 2010

IEEE-SA Standards Board

Approved 4 January 2011

American National Standards Institute

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

The Working Group thanks the International Electrotechnical Commission (IEC) for permission to reproduce Information from its International Publication IEC/TS 62351-3 ed.1.0 (2007).

All such extracts are copyright of IEC, Geneva, Switzerland. All rights reserved. Further information on the IEC is available from www.iec.ch. IEC has no responsibility for the placement and context in which the extracts and contents are reproduced by the author, nor is IEC in any way responsible for the other content or accuracy therein.

Abstract: The DNP3 protocol structure, functions, and application alternatives and the corresponding conformance test procedures are specified. In addition to defining the structure and operation of DNP3, three application levels that are interoperable are defined. The simplest application is for low-cost distribution feeder devices, and the most complex is for full-featured master stations. The intermediate application level is for substation and other intermediate devices. The protocol is suitable for operation on a variety of communication media consistent with the makeup of most electric power communication systems.
Keywords: Distributed Network Protocol (DNP3), distribution automation, distribution feeder, electric power communication systems, master station, substation automation

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA
Copyright © 2010 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1 July 2010. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by the Institute of Electrical and Electronics Engineers, Incorporated.

PDF: ISBN 978-0-7381-6312-3 STD96074
CDROM: ISBN 978-0-7381-6313-0 STDCD96074

IEEE prohibits discrimination, harassment and bullying. For more information, visit http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html.
No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied "**AS IS**."

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation, or every ten years for stabilization. When a document is more than five years old and has not been reaffirmed, or more than ten years old and has not been stabilized, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon his or her independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration. A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal interpretation of the IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, explanation, or interpretation of the IEEE.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Recommendations to change the status of a stabilized standard should include a rationale as to why a revision or withdrawal is required. Comments and recommendations on standards, and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854
USA

Authorization to photocopy portions of any individual standard for internal or personal use is granted by The Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Introduction

This introduction is not part of IEEE Std 1815-2010, IEEE Standard for Electric Power Systems Communications—Distributed Network Protocol (DNP3).

0.1 DNP3 purpose and history

This Introduction discusses the creation and history of DNP3. The structure and operation of the protocol may be easier to understand when taken in the context of the problems the designers of DNP3 intended to solve.

0.1.1 Addressing an impediment to automation

Westronic Incorporated developed DNP3 between 1992 and 1994, intending it to be the first truly open, truly useful protocol standard in the utility industry. Westronic was a manufacturer of remote terminal units and a system integrator based in Calgary, Canada. It had made a reputation converting between the hundreds of proprietary utility protocols in use at the time. This was not an easy task, however, and Westronic management had become frustrated with trying to make its devices compatible with so many proprietary protocols.

A proposal was made that Westronic should develop its own protocol but then release it to the industry. The new protocol would incorporate the best features of the many protocols Westronic had encountered, plus some new ideas. Westronic would place the specification under the control of an independent users' group. Both utilities and vendors would be invited to be members, including Westronic's competitors. Westronic would not receive any money for the sale and distribution of the specification.

0.1.2 Rationale for a new protocol based on standards

Westronic was not the first to propose an open standard for the utility industry, but the designers of DNP3 did not find any of the existing efforts suitable. At the time when Westronic was considering DNP3, there were two main candidates available for an open protocol:

— The Electrical Power Research Institute (EPRI) had recently released the Utility Communications Architecture (UCA) version 1.0. However, version 1.0 listed a choice of protocol profiles only and did not define any object models or services suitable for performing Supervisory Control and Data Acquisition (SCADA) functions. At that point in the development of UCA, very few utilities or vendors had provided input to the specification, and there were some serious concerns about bandwidth usage. These drawbacks and others eventually led to the development of UCA 2.0. UCA 2.0 became an IEEE technical report in 1998 and eventually evolved into IEC 61850.[a]

— The International Electrotechnical Commission (IEC) had developed the first few documents in the IEC 60870-5 series of specifications, including the specifics of the Data Link Layer and general definitions for the Application Layer. (At that time, it was called just 870-5.) Westronic had been participating in this effort but felt that it was progressing too slowly. Furthermore, the IEC had provided many options in the specification, and Westronic was worried the standard would not be restrictive enough to promote interoperability. The IEC eventually released the 60870-5-101-companion standard in 1995 to address these issues.

[a] Information on references can be found in Clause 2.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

In 1992, the IEC work seemed to be the more complete of the two efforts and had wider industry support at the time. Westronic therefore decided to base DNP3 on the IEC work already completed. Even now, the feature sets of IEC 60870-5-101:2003 [B5][b] and DNP3 are very similar because the design teams built them on the same basic research.

UCA was not forgotten. Westronic (by then called Harris Distributed Automation Products) circulated versions of the DNP3 Basic 4 Document Set including a paper called "On the Road to Utility Communications Architecture." The thesis of this paper was that by standardizing on DNP3, utilities would at least be reducing from many protocols to one. This would make it easy for utilities to later change to use UCA. However, very few design elements ofs UCA found their way into this standard, other than a generally layered architecture.

0.1.3 Need for scalability

The designers of DNP3 built it with several goals in mind, but the one that had the most impact on the final protocol was the industry's desire to limit the amount of bandwidth used. At that time, utilities considered a link running at 1200 bits per second to be fairly quick. (Yes, there are areas where this is still true). Local area networks (LANs) were for office computing only, and the thought of trusting one's SCADA network to a third-party telecom provider was heresy.

Power utilities had heard about layered protocols and the Open System Interconnection (OSI) model, but they were unconvinced of their value in a SCADA protocol. The Internet was beginning to boom, of course, but most utilities considered those protocols for business computing only. They were not for a SCADA network. Those who followed such things may also have heard that there was a backlash against the OSI model brewing. Protocols like Asynchronous Transfer Mode (ATM) and Frame Relay promised higher performance by eliminating layers. No utility at that time would have used these protocols in their network, but they probably heard that "layers are bad."

Therefore, the designers of DNP3 gave themselves a design goal to reduce bandwidth and use as few layers as possible.

This goal combined with the desire to be compliant with IEC 60870-5 resulted in the "Transport Function" as it now exists: a header that is not quite part of the Data Link Layer and yet not quite a complete Transport Layer. A later subclause will discuss the Transport Function in more detail.

0.1.4 Emphasis on reliability

While requesting less bandwidth, utilities refused to compromise on the requirement that a SCADA protocol be extremely reliable. Early bit-oriented protocols had acquired a bad reputation because a change of a single bit could result in a device operating the wrong switch. This led to utilities requiring in bid specifications that vendors build select-before-operate, "I tell you twice," functions into all protocols. A few bad experiences made utilities paranoid about reliability to the point of writing it into contracts.

Therefore, when designing a frame format to use, the DNP3 designers chose the most reliable format they could find. The IEC had done extensive modeling on reliability and had documented the results in IEC 60870-5-1:1990 [B3]. Rather than reinvent the wheel, the designers picked the most reliable of the several formats described in that specification, Frame Type 3 (FT3).

In the years that have passed, this decision has proven to be a good one. Many vendors have cursed the calculations necessary for the many cyclic redundancy checks (CRCs). Many system engineers have cursed the extra bandwidth overhead. However, DNP3's reputation for reliability started well and has only improved with the years.

[b] The numbers in brackets correspond to those of the bibliography in Annex E.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

0.1.5 Feature selection

Because the designers of DNP3 were from a systems integration company, they tried to incorporate into DNP3 the best features of all the utility protocols they had encountered. These features included:

— Broadcasting. The ability to send a single message to multiple devices.

— Select-before-operate—or not. The ability to choose to use extra reliability when operating an output, or to choose not to use it.

— Time-stamped data. Some of the most popular utility protocols, such as Modbus, had no way to accurately time-stamp data. Vendors and utilities were forced to develop proprietary work-around solutions. Other protocols supported time stamps on binary data only. DNP3 permits time stamps on almost all data. This is a feature that is only now beginning to become popular as utilities are starting to gather other types of historical data beyond the standard binary "sequence of events" log.

— Accurate time synchronization. Many earlier protocols had no way to account for transmission and software delays when synchronizing. The method used in DNP3 is an amalgamation of several different protocols' solutions.

— Quality flags. Representing a maker of data concentrators, the designers provided a mechanism to see whether data was valid, and why. Some protocols, designed by intelligent electronic device (IED) vendors whose data was always online, did not include this feature.

— Multiple data formats. The ability to report data in a variety of formats: 16-bit, 32-bit, with a flag, without a flag, floating-point, binary-coded decimal (BCD), packed, unpacked, and so on.

— Scan groups. The ability to define and ask for a large set of otherwise unrelated data using a single request.

— Layer separation. Separating the function of "getting the data there" from the actual SCADA functions.

— Report-by-exception. More than any other feature, the ability to reliably report only the changes in data has helped make DNP3 successful.

— Internal indications. As several protocol efforts that are more recent than DNP3 have discovered, it is extremely useful to have a global set of flags returned in each response. These flags indicate the health of the device and the results of the last request.

Most of these features had been seen elsewhere, but this was the first time an open utility protocol had attempted to do them all.

0.1.6 Rationale for DNP3 Subset Definitions

Unfortunately, the "best practices" approach to developing DNP3 was not perfect, causing a number of features to be added that were not really in widespread use. A number of them existed only in Westronic equipment. At various times, vendors have questioned the need for:

— So many different types of counters, particularly delta counters

— So many different types of binary output operations, especially control queuing

— So many different ways to format data (i.e., many qualifier codes)

— Pattern masks

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— Binary-coded decimal analogs

— Storage objects

— The ability to either write or operate an output

— So many layers of confirmation and segmentation

0.1.7 Features to support distributed capabilities

Another trend in the early 1990s was the move to put larger processors and more memory in SCADA devices. Marketing and sales people were talking about "the intelligent network." By this, they meant pushing many of the functions previously performed only by master stations into remote devices. These devices would be more independent and make more decisions on their own. Those who join the utility industry these days are sometimes confused by the term "IED" meaning intelligent electronic device. They say, "Aren't all computing devices intelligent?" Yes, but it wasn't always this way.

In terms of DNP3 design, the idea of "the intelligent network" translated to the following features:

— Spontaneous reporting. A device could transmit whenever it wanted, not just when polled by the master. On multi-drop links, this led to the need for a collision avoidance mechanism.

— Meta-data. The DNP3 designers called a spontaneous message an "Unsolicited Response," which shows the mindset in those days. Most devices only sent data in response to a poll request. Therefore, the master always knew what data was coming. For a device to send an unsolicited message, it had to include not only the SCADA data itself but also information describing the data so the master knew what it was. The term these days for such information is meta-data. It appears in such modern technologies as Extensibe Markup Language (XML). At that time, though, it was a very new concept for the utility industry.

— Wild-carding. Because the remote device was more intelligent, the designers gave it more choice in the amount and format of the data it reported. A master could ask very simple questions, like "Give me all your data" or "Give me your analog changes" and get very complex answers. Again, because the answer did not exactly match the question, meta-data was required in the response.

— Self-description. The idea that a device could tell the master what data it had available, and how to present it, was already around thanks to UCA 1.0. The DNP3 designers tried to incorporate some of this ability into DNP3. The Device Profile Object and the use of floating-point with the units transmitted were considered very advanced. Perhaps they were too advanced because they appeared in very few implementations.

— Vendor-specific expansion. This standard includes the Private Registration Object, which permits vendors to add proprietary extensions to the basic standard. The Private Registration Object Descriptor permits a standard implementation to parse these extensions even though they are proprietary. These objects, too, have not been very popular, but a few vendors have used them to good effect.

— File transfer. The designers gave DNP3 file transfer capabilities so that an intelligent device could download new configuration or software, or upload oscillography files. At the time DNP3 was developed, few devices had flash memory, and only specialized fault recorder devices performed oscillography. Now both are widespread.

— Program control. The ability to start and stop individual programs and processes on a remote device was common in the factory automation industry. DNP3 provides a rudimentary mechanism to do this.

The dream of the "intelligent network" has had mixed results. Some of these features, like spontaneous reporting, meta-data, prioritization and wild-carding, have worked very well. They are probably some of

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

the main reasons for DNP3's popularity. Other features, like self-description, file transfer, floating-point, program control, and collision avoidance, were not completely thought out. The DNP Technical Committee was forced to revise these and issue technical bulletins clarifying their use. Some features have died a death of obscurity.

However, history should not be a harsh judge. Many people take such features for granted these days, but it is important to remember that DNP3 was there first.

0.1.8 Additional communications features

Because of the intense pressure to reduce bandwidth, and because the DNP3 designers had more expertise in SCADA than in general data communications, a number of common communications features were "left out" of the DNP3 definition. Many designers have subsequently mourned the absence of these features. Some of them the DNP Technical Committee has attempted to "add on" afterward. Others the Committee could only achieve now at the cost of obsoleting all existing implementations.

The following list of missing data communications features illustrates how well the DNP3 architecture works despite the limitations imposed at its birth:

— Network layer. At one point, the designers actually wrote a specification for a DNP3 network layer, but Westronic management did not approve it. In retrospect, this is just as well, because the Internet Protocol (IP) network layer now used is far more popular.

— Application Layer addresses. The ability to select a particular logical device within a physical one would have been useful. Most devices that support this feature have found a way around it through local software mechanisms that use the Data Link address and/or physical port number as a key.

— Application and Transport Layer sequence number initialization. This has caused much grief over the years and has been addressed as well as possible without causing obsolescence. Data communications experts should note, therefore, that DNP3 is not quite connection-oriented and not quite connectionless, but somewhere in between.

— Long sequence numbers. DNP3 sequence numbers are very short, which is good for bandwidth but not for detecting duplicates. This is the reason Transmission Control Protocol (TCP) is required when using DNP3 over wide area networks (WANs), which turns out to be a very robust solution.

— Sequence number in Data Link Confirms. Without a sequence number, it is impossible to determine which Data Link frame a Confirm frame is answering. On a serial point-to-point link, this is not a problem, but on a WAN, Confirm frames could arrive out of order or be lost. Using TCP in WANs addresses the issue on IP networks, but in theory, it could still cause problems in serial radio networks. In practice, it generally works anyway. This problem was inherited from IEC 60870-5 and cannot be changed without obsolescence.

— Sliding window. One constant of DNP3 has been that only one transaction can be outstanding at a time. In theory, a device could send several response fragments very quickly for a particular request, but over the years the DNP Technical Committee has decided that interoperability is best served by enforcing a confirmation between each fragment.

— Access security. The designers of DNP3 purposely avoided dealing with this issue because of its complexity. Fortunately, it may be possible to add security features without completely re-writing the protocol.

— Version control. Most protocols tend to have an octet reserved to show the version of the protocol in use. This was not included in the original DNP3 definition due to bandwidth reasons, but it may reappear as part of the new self-description solution.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— Overall length field. Segmentation and fragmentation would have been a lot easier and more robust, and the LAN implementation would have been easier if each fragment had a length field at the beginning. It was not included for bandwidth reasons. Again, various software solutions make it work anyway, so perhaps it was the right decision.

0.1.9 Compatibility with IEC protocols

As discussed earlier, there were two reasons why the DNP3 designers wanted it to be compliant with the IEC 60870-5 specifications:

— They wanted to take advantage of the excellent technical work done on reliability in the IEC 60870-5 Data Link Layer specifications.

— They wanted to increase the acceptance of the protocol by showing it was based on standards work that was already well known.

They were so successful in both efforts that even now some people are confused about whether DNP3 and IEC 60870-5 are interoperable.

The answer is that they are not interoperable, although the DNP3 Data Link Layer could be considered compliant to IEC 60870-5 Parts 1 [B3] and 2 [B4]. DNP3 was based on the drafts available at the time of IEC 60870-5 Parts 1 through 5. These Parts of the specification described the Data Link Layer in great detail and the Application Layer in general. There were several options specified for the Data Link Layer.

The DNP3 designers chose those options of Parts 1 [B3] and 2 [B4] they thought were most appropriate. Unfortunately, when the IEC 60870-5-101:2003 [B5] companion standard was released with the details of the Application Layer, it specified *different* Data Link Layer options than those the DNP3 designers had chosen.

Therefore, DNP3 is considered compliant with IEC 60870-5-1:1990 [B3] and 60870-5-2:1992 [B4] but not with IEC 60870-5-101:2003 [B5].

Table 0.1 shows the differences in the Data Link Layers of the two protocols.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 0.1—Comparison of IEC 60870-5 and DNP3 Data Link Layers

Feature	Options permitted in IEC 60870-5-1:1990 [B3] and 2:1992 [B4]	Chosen by DNP3	Chosen by IEC 60870-5-101:2003 [B5]
Addressing	Single address, length system-dependent	Two-octet Source address and two-octet Destination address. Considered a single four-octet "structured" address for compliance purposes.	Single address, choice of either zero, one, or two octets in length
Frame Format	Choice of FT1.1, FT1.2, FT2, and FT3	FT3, transmitted asynchronously	FT1.2
Reliability Mechanism	Varies per frame type	Multiple 16-bit CRCs over each 16 octets of a 255-octet frame. Start and Stop bits, but no parity.	Parity bits and one-octet checksum (not CRC) calculated over 255 octets
Hamming Distance	Varies per frame type	6 for the original FT3. Some debate about the value as currently used. See further discussion in this clause.	4
Acknowledge-ments	Either fixed-length or single-octet	Fixed 10-octet only	Either fixed-length or single-octet
Procedures	Balanced (no master) or Unbalanced (master polls)	Balanced only	Either Balanced or Unbalanced
Method for Multi-Drop Links	Unbalanced mode	Collision Avoidance	Unbalanced mode

0.1.9.1 Hamming Distance

Some critics of DNP3 have disputed DNP3's right to claim a Hamming Distance of six. The "Hamming Distance" of a protocol is the number of bit errors required in a frame before a receiver could incorrectly identify a corrupted incoming frame as a valid frame. Critics argue that the original calculation was made assuming the FT3 frame was transmitted synchronously, while DNP3 uses the FT3 frame format asynchronously.

The main concern in this debate is inter-character gaps. If a gap is permitted between the octets of a 16-octet block, noise could be introduced that might be misinterpreted as valid data. In fact, it has been shown that there exists at least one case where an inter-character gap of exactly 1-bit time at the end of a message can be misinterpreted, thereby resulting in a Hamming distance of 2. Critics claim that this standard has never *required* that all octets of a block be transmitted together, and this reduces the theoretical reliability of the protocol to below that of the FT1.2 frame.

However, years of use in hundreds of systems have proven DNP3's reliability to be more than sufficient for utility purposes. This may be due to the fact that most DNP3 devices transmit frames without inter-character gaps, and receiving devices tend to start a timer or other mechanism that discards incoming frames when inter-character gaps appear.

The inter-character concern with the DNP3 frame is similar to a problem that occurs in some IEC 608705-101 systems. The FT1.2 frame's reliability relies on the use of parity bits in each octet. However, many utilities mistakenly use the protocol with modems that do not add, or actually remove, such parity bits. The IEC is preparing an IEC 60870-5 standard that clarifies parity bits *shall* be used.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

0.1.9.2 Addressing of binary outputs

The other main issue concerning DNP3 compliance to IEC 60870-5 was the structure of the address field. The IEC definition of the address field states that it is a single address, always addressing one end of the link. This is the way IEC 60870-5-101:2003 [B5] uses the address field.

By including both a source and a destination in every message, the DNP3 designers permitted the use of multiple masters on the same link, and peer-to-peer communications. This proved to be a powerful argument in the acceptance of DNP3. Furthermore, since IEC 60870-5-2:1992 [B4] did not specify a particular length of address, a four-octet address that just happened to be "structured" with two sub-addresses could still be considered compliant.

0.1.9.3 Reality today

Although it was the topic of lively debate when DNP3 was first released, the question of whether DNP3 complies with IEC 60870-5 is essentially a moot point today. DNP3 may be considered compliant to Parts 1 and 2. One could even argue that DNP3 complies with the spirit, if not the letter, of Part 5, the general Application Layer definition. However, the format of the IEC 60870-5-101 [B5] Application Layer is very different from that of DNP3. It is clear the two protocols could never interoperate.

It is better to consider the two protocol suites as cousins with a common family tree and leave it at that.

0.1.10 Transport Function

The naming of the Transport Function always confuses newcomers to DNP3. Is it a true Transport Layer, is it a part of the Data Link Layer, or is it something truly different?

The answer is that it really is something different, although it most closely resembles an additional field in the Data Link Layer. It does not have its own addressing or acknowledgments, as a separate layer would. There was no network layer in the original protocol definition, so the transport header was terminated at the end of each physical link, just like the data link header. It does not have the long sequence numbers and other features that would really enforce transmitting frames in sequence. Therefore, it does not seem to be a Transport Layer.

However, if it were a field of the data link header, it would be included in every data link frame, and it is not. Only those frames containing Application Layer data contain a transport header.

The reasons this strange "half-layer" exists are both political and technical. The designers of DNP3 decided they wanted the Application Layer data broken into small segments suitable for passing over noisy links. This capability would at a minimum require a new Data Link Layer field. However, they did not want to add a new field for two reasons:

a) It would eliminate DNP3's chances to be considered compliant to IEC 60870-5. As discussed earlier, this was considered critical to DNP3's acceptance by the industry.

b) Changing the structure of the FT3 frame could possibly compromise the calculated reliability of the frame.

Therefore, the transport header was placed in front of the Application Layer header, in the user data field of the Data Link Layer frame.

However, the next question was, "What to call it?" As noted in this clause, it had some of the characteristics of a layer but not all of them. Furthermore, the designers knew there would be resistance to any additional layers in the protocol. It was bad enough that they were dedicating *a whole octet* to the segmentation and reassembly functions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Therefore, the name "Transport Function" was chosen, thus causing years of questions on hotlines, e-mails, and training presentations.

Whatever it is, it makes DNP3 distinct. Along with Application Layer fragmentation, it permits a small, low-powered device to report a nearly unlimited amount of data reliably over a noisy link.

0.1.11 DNP User's Group

One feature of DNP3 that newcomers do not always appreciate is the organization that stands behind it. Over the years, the DNP User's Group has contributed at least as much to the protocol's success as the technical features of the protocol itself.

In roughly chronological order, here are the organizational features that helped DNP3 become popular:

- Membership that included both utilities and vendors
- Low membership fees
- Low cost of the specifications
- A structure consisting of steering, technical, and marketing committees
- The DNP3 Subset Definitions document
- Suggested wordings for utilities to specify DNP3
- Agreement that any non-backward compatible change shall be approved by the General Membership
- The DNP3 hotline, later to become an Internet chat session
- Booths at major trade shows
- Publishing membership lists so utilities could see the lists of vendors
- The DNP3 bulletin board, later to become a Web site
- The DNP3 e-mail mailing list
- The DNP Technical Committee mailing list, which can include non-committee members
- Publishing the technical committee minutes so the process remains open
- Technical bulletins clarifying areas of dispute when interoperability issues arose
- Agreement, first informal and then formal, that the president should always be from a utility
- The DNP3 IED Application Level Conformance Test Documents[c]
- The DNP3 WAN/LAN Specification

0.1.12 Summary

The following design goals, whether formally stated or not at the time, had a major impact on the structure of DNP3:

[c] Refer to htp;//www.dnp.org.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— Include the best features of the utility protocols in use at the time.

— Push the intelligence in the network toward the remote device.

— Try to comply as much as possible with existing standards efforts, especially IEC 60870-5.

— Use as little bandwidth as possible.

— Make it more reliable than anything that came before.

It is easy to see that these goals are necessarily contradictory. The resulting protocol was not perfect and has been "patched up" over the years. However, it remains popular, open, reliable, and mostly backward-compatible.

0.1.13 Background: Origins of the name "DNP3"

Few people seem to know what to call this protocol. Everyone knows it is DNP3, but is it "DNP 3," "DNP 3.0," "DNP V3.0," or any combination of the above? Also, there are several different subset levels of implementation, and a few non-backward-compatible changes have been made over the years.

As of Technical Bulletin TB2000-003: "Change Management," the official name of the protocol is DNP3-xxxx, where xxxx is the year of release of the Test Procedures to which a device complies. The subset level is specified afterward, as in "DNP3-2000 Level 2."

This naming convention represents an evolution of the name over the years:

— The original Basic 4 documentation referred to the protocol as "DNP V3.00." No one has ever liked saying the "V" part, so that name has never caught on.

— For those who were wondering: DNP V1.00 and DNP V2.00 are proprietary Westronic protocols that were rarely used even at the time DNP3 was released.

— The user's group is called just the "DNP User's Group." That saves it from having to worry about version numbers in marketing information.

— The Subset Definitions defined the format DNP3-Lx, where x was the subset level. That never caught on either, since utilities preferred to spell out the words "Level x" in their bid specs.

— When the Test Procedures were first published in 2000, there had to be some mechanism to distinguish between an implementation that was compliant to the procedures and earlier implementations that were not. The DNP Technical Committee therefore decided to use the year in the specification, similar to the format used by the International Organization for Standardization (ISO), IEEE, and IEC.

— The intent is that there shall never be a DNP 4.0, or even a DNP 3.1. To help illustrate this commitment to backward compatibility, the DNP User's Group changed the name from "DNP V3.00" to "DNP3." The name therefore remains recognizable while eliminating the "software version" impression that the decimal point gave.

Some people rightly complain that it is redundant to say "DNP3 protocol" since the "P" in DNP3 stands for "Protocol" already. However, this truism does not seem to discourage people from using the phrase, and it is likely to be heard for years to come.

Now if one could just figure out how a protocol that originally had no network layer ended up with the name "Distributed *Network* Protocol." One long-standing DNP3 user points out that the networks in question are SCADA networks, which "bear scant resemblance to other things that people usually call networks." Perhaps this is the case.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Ah well, a protocol by any other name is just as interoperable and reliable.

0.2 DNP3 overview

0.2.1 Basic messages and data flow

The document is a brief, but incomplete, overview of DNP3 messages and data flow. Its purpose is to prepare the reader for what follows in the Application Layer, Transport Function, and Data Link Layer specifications for DNP3.

This initial discussion of DNP3 uses the master–outstation model illustrated in Figure 0.1. This clause omits many details to purposely keep the description straightforward.

Figure 0.1—DNP3 master–outstation model

The User Layer in the master on the left side of the figure initiates a data transfer by causing its Application Layer to send a request to the outstation. The request contains a function code and zero or more DNP3 objects that specify what data is wanted. The Application Layer passes the request on to the Transport Function for partitioning into transmission-sized units and then on to the Data Link Layer. The Data Link Layer adds addressing and error detection information and transmits the packet to the outstation over the physical media.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

At the outstation on the right side, the Data Link Layer receives the octets from the physical layer and checks for errors that were introduced while the packet was in transit. If no errors are detected, the addressing and error detection information added by the transmitting Data Link Layer is stripped from the message, and the remaining octets are passed on to the Application Layer. If necessary, the Transport Function reassembles multiple packets into a complete request. The Application Layer then interprets the function code and DNP3 objects in the message and indicates to the User Layer what data is desired.

The User Layer in the outstation initiates a response based on what data the master requested. It fetches data, classifies it, and presents that data to the Application Layer. The Application Layer creates a message with data formatted into DNP3 objects, passes it through the Transport Function, and then on to the Data Link Layer for transmission to the master using methods similar to those employed by the master to send its request.

Upon receipt of the response at the master, the layers perform address and error checking and reassembly into a complete message for the Application Layer. This layer parses the DNP3 objects in the response and presents the information to the User Layer. The User Layer can then store or operate on that data in a way that is suitable for the end user.

The master always initiates control commands. These actuate device outputs or variables internal to the outstation. The DNP3 User-to-Application Layer interface and transmission procedures are similar to those discussed for data acquisition.

A transaction consists of a single request followed by a single response. A master sends a request and waits for the response, or a timeout, before issuing another request. Multiple transactions may simultaneously occur within a system. For example, consider the case where two masters each make requests to the same outstation.

In some systems the master does not always directly initiate data transfer. DNP3 has provision for the outstation to automatically send data when it detects a condition worthy of transmitting without a specific master request. “Unsolicited responses” is the terminology applied to this type of operation because the request is implied.

0.2.2 Layering

0.2.2.1 General

ISO defines a communication architecture that separates functions into seven layers called the OSI reference model. DNP3 protocol is based on a simplified model termed the Enhanced Performance Architecture (EPA) that consists of only three layers: Application, Data Link, and Physical. Figure 0.1 shows how DNP3 fits the EPA structure and communication model.

In theory, each layer in a layer stack performs a set of functions required to communicate with the same layer in another device, relying on the next lower layer for more primitive functions. At the sending device, each layer below the Application Layer receives data from the layer above for transmission. The layer adds more information that enables the equivalent layer in the receiver to properly process the message. At the receiving device, layers examine their layer specific information added by the corresponding layer at the transmission site and process the message appropriately. The layer control information is stripped, and the message is passed to the next higher layer.

The Transport Function within the Application Layer performs a layer-like function of partitioning large messages into smaller messages that the Data Link Layer is capable of handling. The Transport Function is sometimes referred to as a “pseudo layer.” In DNP3 the Application Layer, Transport Function, and the Data Link Layer in the transmitter add information to the message for enabling the same layer or pseudo layer in the receiver to process the message.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

0.2.2.2 Fragments, segments, and frames

Figure 0.2 illustrates the partitioning of large messages at the Application Layer into smaller units and the addition of header information at each layer.

Figure 0.2—Fragmented Application Layer message

Figure 0.2 shows a fragmented Application Layer message, segmentation of each fragment by the Transport Function, and how segments fit into Data Link Layer frames. This diagram does not show timing and confirmation details but serves to demonstrate how the higher level parts nest inside the lower layer structures. It also shows the relative positions of the Application Layer headers, the Transport Function headers, and the Data Link Layer headers.

Table 0.2 provides a summary of the terminology and some brief information associated with each layer or function.

Table 0.2—DNP3 layer summary

Layer or function	Unit name	Information
Application Layer	Application Fragment	Permits the setting of an upper limit on the memory requirements for message reception. Requests shall fit into a single fragment. Responses may require more than one fragment.
Transport Function	Transport Segment	Segmentation breaks a Data Link fragment into pieces that fit into a Data Link frame. Each segment contains a Transport header, but only the first segment of any fragment contains an Application header. Each segment may have a maximum of 250 octets including the Transport header.
Data Link Layer	Data Link Frame	A Frame may have as many as 292 octets including its header and CRC octets. Frames are designed for superior error detection.

0.2.3 Message sequences

Figure 0.3 illustrates a hypothetical sequence and the time relationship of fragments and frames as they move between layers, and between the master and the outstation in a **polled** environment. Readers just beginning to learn this standard are cautioned to only view the diagram as a means of gaining a general overview.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 0.4 illustrates a hypothetical sequence and the time relationship of fragments and frames as they move between layers, and between the master and the outstation in an **unsolicited** response environment. Readers just beginning to learn this standard are cautioned to only view the diagram as a means of gaining a general overview. Later, after studying the details, refer back to this figure when it may be more meaningful.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 0.3—Polled sequence with link layer confirmation

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 0.4—Unsolicited response sequence

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

0.2.4 Data loss and efficiency

One of the fundamental goals of DNP3 is to prevent loss of data transferred from an outstation to the master. Of special concern is the transfer of all binary input states, in sequence, and without missing any transitions.

To increase the efficiency, DNP3 provides for report-by-exception whereby changes are transmitted soon after they occur, and an occasional integrity poll is issued to synchronize the master and outstation databases. When an outstation transmits changes, it shall request Application Layer confirmation. Only after the master confirms receipt of the changes can the outstation assume the changes arrived at the master.

Outstation devices that are able to report all of their current data in a single frame are not required to support report-by-exception.

0.2.5 Unsolicited responses

Unsolicited responses are messages spontaneously sent from an outstation without a specific request from a master when "something of significance" occurs. The DNP3 protocol includes support for unsolicited responses.

This method of operating has advantages in some applications. In a system with a large number of outstations and a single master, changes at an outstation can reach the master often much faster because there is no delay while waiting for a master poll. The communication costs to achieve faster polling in some installations can be prohibitive, and the quickest notification of changes can occur if most of the messages contain only changes and confirmations. Unsolicited operation may reduce costs where the owners choose a "cost-per-byte" type of service.

On the other hand, equipment that implements unsolicited messages is more complex because the issues of media access and collision avoidance should be dealt with. Master software requires accepting messages from any of its outstations at any time. Another disadvantage is that system performance may become unpredictable during periods of heavy communication.

Employing unsolicited reporting requires an engineering judgment based on numerous factors for each individual system. There are no guarantees that unsolicited reporting is universally applicable for all systems.

A description and rules for unsolicited responses are provided in Clause 4 through Clause 6 of this standard.

0.2.6 IP networking

DNP3 was originally designed for serial octet streaming, point-to-point communications over voice grade audio links, or hard-wired, multi-drop wire and fiber cabling. As IP networking evolved, users recognized a need for devices to exchange DNP3 messages over these high-speed, packet-based, digital networks. DNP3 now includes this capability.

The approach taken was to place DNP3 as the user of an IP stack and to retain all the same Application Layer, Transport Function, and Data Link Layer structures, objects, and formats as in the original DNP3. Thus, DNP3 Data Link Layer frames are passed transparently across an IP network as TCP or User Datagram Protocol (UDP) packets.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A description and special rules are described in Clause 13 of this standard.

0.3 Organization of DNP3 Specification

The complete DNP3 Specification is organized into separate clauses wherein details of the DNP3 protocol are documented as follows[d]:

Clause 0 through Clause 3 and Annex B:	DNP3 Introduction (formerly known as Volume 1)
Clause 4 through Clause 7:	Application Layer (formerly known as Volume 2)
Clause 8:	Transport Function (formerly known as Volume 3)
Clause 9:	Data Link Layer (formerly known as Volume 4)
Clause 10:	Layer Independent Topics (formerly known as Volume 5)
Clause 11, Clause 12, and Annex A:	Data Object Library (formerly known as Volume 6)
Clause 13 and Annex C:	IP Networking (formerly known as Volume 7)
Clause 14:	Interoperability (formerly known as Volume 8)

0.4 Conventions used in this standard

0.4.1 NOTEs

Some NOTEs appear inside a box.[e]

NOTE

> This is a NOTE box. These are used to highlight special information that is not part of the protocol specification but can help the reader.
>
> NOTE boxes also hold implementation suggestions.

0.4.2 Examples

Examples are preceded with a box describing what is illustrated below. The number EX 1 represents the example number.

EX 1	This example shows a request for all of the static binary inputs. Assume there are 18 binary inputs.

[d] The DNP3 Specification was identified by volumes prior to IEEE Std 1815-2010 standardization.
[e] Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

0.4.3 Single master, single outstation perspective

The DNP3 protocol is suitable for systems with one or more master stations, one or more outstations, and peer-to-peer arrangements. In general, this specification was written from the perspective of a single master and a single outstation to make the documents easier to understand without the additional complexities involved.

A separate subclause (13.2.3.5) is devoted to discussion of multi-master systems and their special considerations and requirements. Statements appear elsewhere only when it is necessary to emphasize specific characteristics or behavior for systems with multiple master or outstation devices.

Notice to users

Laws and regulations

Users of these documents should consult all applicable laws and regulations. Compliance with the provisions of this standard does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

This document is copyrighted by the IEEE. It is made available for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making this document available for use and adoption by public authorities and private users, the IEEE does not waive any rights in copyright to this document.

Updating of IEEE documents

Users of IEEE standards should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect. In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE Standards Association web site at http://ieeexplore.ieee.org/xpl/standards.jsp, or contact the IEEE at the address listed previously.

For more information about the IEEE Standards Association or the IEEE standards development process, visit the IEEE-SA web site at http://standards.ieee.org.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: http://standards.ieee.org/reading/ieee/updates/errata/index.html. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: http://standards.ieee.org/reading/ieee/interp/index.html.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Participants

At the time this standard was submitted to the IEEE-SA Standards Board for approval, the Standard for Electric Power Systems Communications—Distributed Network Protocol (DNP3) Working Group had the following membership:

H. Lee Smith, *Chair*
Andrew West, *Vice Chair*

Bill Ackerman
Philip Aubin
Joey Baranski
Alan Bishop
Jake Brodsky
Andreou Demos
Mike Dood
Ron Farquharson
Chris Francis
Marc Lacroix
Thomas Man
Parker McCauley
Bruce Muschlitz
Craig Preuss
James Recchia
Craig Rodine
William F. Schmid
Samuel Sciacca
Alan Scott
Barry Shephard
Michael S. Smith
John T. Tengdin
Eric Thibodeau
Jay Vellore
David Wood
Tianling Wu

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Bill Ackerman
S. Aggarwal
Carlo Arpino
Ali Al Awazi
John Banting
Hugh Barrass
Jeffrey Benach
Gabriel Benmouyal
H. Stephen Berger
Martin Best
Wallace Binder
Paul Bishop
Steven Brockschink
Gustavo Brunello
William Byrd
Arvind K. Chaudhary
He Chun
Michael Coddington
Stephen Conrad
R. W. Corlew
Alireza Daneshpooy
Em Delahostria
Gary L. Donner
Mike Dood
Randall Dotson
Michael Edds
Ahmed Elneweihi
Gary Engmann
Herbert Falk
Ron Farquharson
Fredric Friend
James Gardner
Ramez Gerges
David Geigel
Jalal Gohari
Edwin Goodwin
Stephen Grier
Randall Groves
Ajit Gwal
John Harauz
Edrawd Hare
David Harris
Gary Heuston
Gary Hoffman
Dennis Holstein
David Horvath
C. Huntley
Hachidai Ito
R. Jackson
Junghoon Jee
Lars Juhlin
Innocent Kamwa
John Kay
Chad Kennedy
Gael Kennedy
Tanuj Khandelwal
Yuri Khersonsky
Morteza Khoidaie
Stanley Klein
Hermann Koch
Joseph L. Koepfinger
Benjamin Kroposki
John Kueck
Jim Kulchisky
Marc Lacroix
Chung-Yiu Lam
Frank Lambert
Richard Lancaster
Daniel Lubar
G. Luri
Faramarz Maghsoodlou
Kenneth Martin
James McConnach
John McDonald
David McGinn
Gary McNaughton
Willam Moncrief
Jose Morales
Kimberly Mosley
Jerry Murphy
R. Muprhy
Bruce Muschlitz
Pratap Mysore
Anthony Napikoski
Ronda Netzel
Michael S. Newman
Gary Nissen
Satoshi Obara
Donald Parker
Bansi Patel
M. Pehosh
Bruce Pickett
Louis Pinto
Craig Preuss
John Randolph
R. Ray
Michael Roberts
Charles Rogers
M. Sachdev
Bob Saint
Steven Sano
Bartien Sayogo
Samuel Sciacca
Douglas Seely
Devki Sharma
Gil Shultz
Cheong Siew
Mark Simon
H. Lee Smith
Jerry Smith
Aaron Snyder
John Spare
Gary Stoedter
James Swank
Ricahrd Taylor
William Taylor
John T. Tengdin
David Tepen
Elisabeth Tobin
Sylvester Toe
Joseph Tumidajski
Joe Uchiyama
Eric Udren
John A. Vandermaar
John Vergis
Ilia Voloh
Daniel Ward
Kenneth White
Thomas Wier
Peter Wong
Larry Young

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

When the IEEE-SA Standards Board approved this standard on 17 June 2010, it had the following membership:

Robert M. Grow, *Chair*
Richard H. Hulett, *Vice Chair*
Steve M. Mills, *Past Chair*
Judith Gorman, *Secretary*

Karen Bartleson
Victor Berman
Ted Burse
Clint Chaplin
Andy Drozd
Alexander Gelman
Jim Hughes
Young Kyun Kim
Joseph L. Koepfinger*
John Kulick
David J. Law
Hung Ling
Oleg Logvinov
Ted Olsen
Ronald C. Petersen
Thomas Prevost
Jon Walter Rosdahl
Sam Sciacca
Mike Seavey
Curtis Siller
Don Wright

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Satish Aggarwal, *NRC Representative*
Richard DeBlasio, *DOE Representative*
Michael Janezic, *NIST Representative*

Don Messina
IEEE Standards Program Manager, Document Development

Matthew J. Ceglia
IEEE Standards Program Manager, Technical Program Development

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Contents

1. Overview 1
1.1 Scope 1
1.2 Purpose 2
1.3 Octet order 2

2. Normative references 3

3. Definitions, acronyms, and abbreviations 5
3.1 Definitions 5
3.2 Acronyms and abbreviations 9
3.3 Special terms 12

4. Application layer—part 1 13
4.1 Application layer preface 13
4.2 Message structure 19
4.3 Fragment rules 39
4.4 Detailed function code procedures 41
4.5 Detailed IIN bit descriptions 85
4.6 Unsolicited responses 93

5. Application layer—part 2 106
5.1 Additional details 106
5.2 Using virtual terminal objects 114
5.3 Sequential file transfer 117
5.4 Data sets 125
5.5 Device attributes 153

6. Application layer—part 3 160
6.1 State tables and diagrams 160

7. Secure authentication 175
7.1 Introduction 175
7.2 Description of operation 178
7.3 Formal specification 188
7.4 Interoperability requirements 216
7.5 Special applications 219
7.6 Compliance with IEC/TS 62351-3 223
7.7 Compliance with IEC/TS 62351-5 226

8. Transport function 229
8.1 Overview 229
8.2 Transport function description 230

9. Data Link Layer 236
9.1 Overview 236
9.2 DNP3 Data Link Layer description 236
9.3 State tables and diagrams 249
9.4 Sample CRC calculations 259

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

10. Layer-independent topics 263
10.1 Purpose of layer-independent topics 263
10.2 Confirmation and retry guidelines 263
10.3 Time synchronization 266
10.4 Handling multiple messages 273

11. Data object library 275
11.1 Data object library—part one 275

12. DNP3 object library—parsing codes 321
12.1 Data object library—part three 321

13. IP networking 361
13.1 IP networking overview 361
13.2 Layer requirements 362
13.3 Security 376
13.4 Time synchronization 377
13.5 UML statecharts 377

14. Interoperability 379
14.1 About this clause 379
14.2 Overview 380
14.3 Level 1 DNP3 implementation (DNP3-L1) 383
14.4 Level 2 DNP3 implementation (DNP3-L2) 388
14.5 Level 3 DNP3 implementation (DNP3-L3) 393
14.6 Level 4 DNP3 implementation (DNP3-L4) 403
14.7 Conformance 417
14.8 XML representation 418
14.9 Instructions for creating a Device Profile Document 427

Annex A (normative) DNP3 object library 429

Annex B (informative) DNP3 quick reference 736

Annex C (informative) Associations 741

Annex D (normative) UTF-8 related copyright 744

Annex E (informative) Bibliography 745

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

IEEE Standard for Electric Power Systems Communications—Distributed Network Protocol (DNP3)

IMPORTANT NOTICE: This standard is not intended to ensure safety, security, health, or environmental protection. Implementers of the standard are responsible for determining appropriate safety, security, environmental, and health practices or regulatory requirements.

This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading "Important Notice" or "Important Notices and Disclaimers Concerning IEEE Documents." They can also be obtained on request from IEEE or viewed at http://standards.ieee.org/IPR/disclaimers.html.

1. Overview

1.1 Scope

This document specifies the DNP3 protocol structure, functions, and application alternatives. In addition to defining the structure and operation of DNP3, the standard defines three application levels that are interoperable. The simplest application is for low-cost distribution feeder devices, and the most complex is for full-featured master stations. The intermediate application level is for substation and other intermediate devices. The protocol is suitable for operation on a variety of communication media consistent with the makeup of most electric power communication systems.

The standard consists of several clauses each related to an application or function.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

1.2 Purpose

The purpose of this standard is to document and make available the specifications for the DNP3 protocol. While a primary focus of this protocol is the Electric Utility Industry, other industries that deliver Energy and Water are also using DNP3. The intent of this DNP3 standard is to meet the goal established by the National Institute of Standards and Technology (NIST) for a Smart Grid protocol:

- Provides a protocol standard from a recognized standard institution
- Provides interoperability with hundreds of operational systems and thousands of devices
- Provides cyber security based on IEC/TS 62351-5
- Provides Device data profiles in a format that can be mapped to IEC 61850 Object Models

Vendors may use this standard to implement and test the protocol in their products and be assured of interoperability. Users may use the document to specify the features they wish to apply. System Integrators may use this standard to assist in system integration and testing.

1.3 Octet order

Unless specified elsewhere, the least significant octet in multi-octet data values is transmitted first.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

FIPS 186-2, *Digital Signature Standard (DSS)*, USA NIST, February 2000 including Change Notice #1, October 2001.[1, 2]

FIPS 198, The Keyed-Hash Message Authentication Code (HMAC), March 2002.

IEC 60870-5, Telecontrol equipment and Systems—Part 5: Transmission protocols.

IEC 61850, A New Approach to Substation Automation,Communications, and Integration.[3]

IEC/TS 62351-2, Power systems management and associated information exchange—Data and communications security. Part 2: Communication network and system security—Glossary.

IEC/TS 62351-3, Power systems management and associated information exchange—Data and communications security. Part 3: Communication network and system security—Profiles including TCP/IP.

IEC/TS 62351-5, Power systems management and associated information exchange—Data and communications security. Part 5: Communication network and system security—Security for IEC 60870-5 and derivatives.

IEEE Std 754™, IEEE Standard for Floating-Point Arithmetic.[4, 5]

IETF RFC 768, User Datagram Protocol.[6]

IETF RFC 791, DARPA Internet Program Protocol Specification.

IETF RFC 793, Transmission Control Protocol DARPA Internet Program Protocol Specification.

IETF RFC 3280, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile

IETF RFC 3629, UTF-8, a transformation format of ISO 10646

---

[1] FIPS publications are available from the National Technical Information Service (NTIS), U. S. Dept. of Commerce, 5285 Port Royal Rd., Springfield, VA 22161 (http://www.ntis.org/).

[2] Only the random number generation algorithms in the appendix are used.

[3] IEC publications are available from the Sales Department of the International Electrotechnical Commission, Case Postale 131, 3 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (http://www.iec.ch/). IEC publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

[4] The IEEE standards or products referred to in this clause are trademarks owned by the Institute of Electrical and Electronics Engineers, Incorporated.

[5] IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).

[6] IETF publications are available from the Internet Engineering Task Force, IETF Secretariat, c/o Association Management Solutions, LLC (AMS), 48377 Fremont Boulevard, Suite 117, Fremont, CA 94538 (http://www.ietf.org).

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

ISO/IEC 9798-4, Information technology—Security techniques—Entity authentication—Part 4: Mechanisms using a cryptographic check function.[7]

ISO/IEC 10646, Information technology—Universal Multiple-Octet Coded Character Set (UCS).

[7] ISO publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/ Suisse (http://www.iso.ch/). ISO publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA (http://www.ansi.org/). IEC publications are available from the Sales Department of the International Electrotechnical Commission, Case Postale 131, 3 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (http://www.iec.ch/). IEC publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. *The IEEE Standards Dictionary: Glossary of Terms & Definitions* should be referenced for terms not defined in this clause.[8]

Application Service Data Unit (ASDU): The data presented by the Application Layer to lower layers for transmission. Equivalent to a Distributed Network Protocol (DNP3) Application Layer fragment.

association: The state information required to maintain a logical connection between a master and an outstation. Includes both the third-generation Distributed Network Protocol (DNP3) and the Internet Protocol information.

challenger: A station that issues authentication challenges. May be either a master or an outstation.

configure: To initialize a device so it operates in a particular way. For instance, a customer may *configure* a device so the device never requests data link confirmations. A vendor or customer may configure a device using a variety of mechanisms (e.g., parameters in non-volatile random access memory (NVRAM), parameters in read-only memory (ROM), dip switches, or hardware jumpers).

control direction: Data transmitted by the master to the outstation.

control relay output block (CROB): A structured data block appearing in request and response messages associated with actuating on-off type output devices.

cyclic redundancy check (CRC): Code that is generated according to a specific algorithm, and transmitted with the message, for the purpose of detecting data corruption during communication via the Physical Layer.

datagram end point: A process that can send and receive messages using the connectionless User Datagram Protocol (UDP).

deprecated: Indicates that a feature or requirement is still permitted although its use is discouraged, and it is not guaranteed to be a part of future specification versions. *See also:* **obsolete**.

Distributed Network Protocol (DNP3) event object: A third-generation DNP3 object that has a group number and variation that is used to report event data in the outstation to the master.

Distributed Network Protocol (DNP3) object: A third-generation DNP3 object that is a representation of data from a point or other structure that is formatted according to its group and variation numbers for transport in a message.

dual end point: A process that can both listen for Transmission Control Protocol (TCP) connection requests and perform a TCP active open on the channel. A dual end point in an outstation shall also receive User Datagram Protocol (UDP) datagrams to facilitate broadcast requests.

end point: A process on a computer that accepts and/or initiates communication channel establishment and provides a service for other processes to send messages through the channel.

[8] *The IEEE Standards Dictionary: Glossary of Terms & Definitions* is available at http://shop.ieee.org/.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

event: The occurrence of something significant happening.

event data: The information that is retained regarding an event. Event data are saved at the outstation in vendor-specific structures and reported to the master using third-generation Distributed Network Protocol (DNP3) event objects. Event data shall be kept in the outstation until confirmation has been received from the master indicating that a description of the event arrived at the master. Only after confirmation, may the outstation discard the corresponding event data. With a few exceptions, DNP3 does not define which events are worthy of transmission.

event buffer: A vendor-specific structure used to hold event data in an outstation. Third-generation Distributed Network Protocol (DNP3) does not specify how many buffers an outstation maintains. When the plural buffers is used in this specification, it also refers to the single common structure that some implementation have.

fragment: A packet of octets that is sized to fit into the buffers of the receiving device's Application Layer. Each fragment has an application header and octets containing Application Layer request, response, or confirmation information.

frame: A packet of octets transmitted from the Link Layer in one device to the Link Layer in another device over the Physical Layer. Each frame contains a link header, cyclic redundancy check (CRC) octets, and sometimes a segment from the Transport Function.

identical retry. An *identical retry* of an unsolicited response is a repeat, octet-for-octet of the previously transmitted unsolicited response. The internal indications (IIN) octets shall be the same. The sequence number appearing in the application control octet of an identical retry and the previous unsolicited messages are the same.

initiating end point: An initiating process that performs a Transmission Control Protocol (TCP) active open on the channel.

input: Refers to values that are measured, read, or generated by the device and are reported by an outstation to a master. Examples are

— The level of fluid in a tank

— The open-close state of a switch

— The calculated sum of the power on all three phases of a power line

Input sometimes refers to the physical source of the value such as a voltage sensor.

internal indications (IIN): This bit field appears in response headers that indicate certain states or error conditions with the outstation.

Least Significant Byte (LSB): The third-generation Distributed Network Protocol (DNP3) uses the term *octet* instead of *byte*; therefore, this abbreviation means the least significant octet. It is applied when there are two or more contiguous octets that together are used to hold a value and the lower order octet is intended.

listening end point: A listening process that performs a Transmission Control Protocol (TCP) passive open and waits for connection requests. A listening end point shall also receive User Datagram Protocol (UDP) datagrams to facilitate broadcast requests.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

local issue or local matter: The subject of interest that is restricted to an individual device or system and not generally known to other devices, systems, vendors, or persons. The method of measuring analog quantities in an outstation is a local issue.

local mode: An operating condition whereby outputs are prevented from being controlled remotely from a third-generation Distributed Network Protocol (DNP3) master. The outputs can possibly be operated locally at the device where the output point is physically located. Output points that are not configured for operation by DNP3 are not considered to be in local mode.

master: A device that initiates requests to gather data or perform controls using the Distributed Network Protocol (DNP3).

message: A block of data that is part of the authentication mechanism. It shall be expressed by some combination of Distributed Network Protocol (DNP3) objects, function codes, qualifiers, and so on.

monitoring direction: Data transmitted by the outstation to the master.

Most Significant Byte: The third-generation Distributed Network Protocol (DNP3) uses the term *octet* instead of *byte*; therefore, this abbreviation means the most significant octet. It is applied when there are two or more contiguous octets that together are used to hold a value and the higher order octet is intended.

network address translation (NAT): An Internet standard that enables a local area network (LAN) to use one set of Internet Protocol (IP) addresses for internal traffic and a second set of addresses for external traffic.

null response: A response message wherein the Application Layer fragment consists of only application control, function code, and internal indication octets.

obsolete: Indicates that a feature or requirement is no longer valid; vendors shall not implement them, and users shall disregard any past references in previous specification documents. *See also:* **deprecated**.

octet: A group of eight contiguous digital information bits.

original. The first message in an unsolicited response series. The sequence number in the application control octet is incremented from the previously transmitted unsolicited response. *See also:* **unsolicited response series**.

output: Refers to values in an outstation or lower level device that are controlled by commands from the master. Examples are

— An analog signal that sets the desired pressure for a gas manifold

— Electrical contacts, which when activated, cause a circuit breaker to trip or close

Output sometimes refers to the physical device that receives a control signal such as a circuit breaker.

parse: To resolve a request or response into component parts. In the context of third-generation Distributed Network Protocol (DNP3) messages, to *parse* a message means a device can break the message into pieces, each of which consists of a header and sometimes some corresponding data. If a device is able to parse a message, it can recognize each piece of a message. It does not necessarily make use of the data found in that message. However, it shall make any confirmation responses or other responses that the message requires.

point: An instance of a point type.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

point index: The zero-based numeric identifier that differentiates unique instances of points having the same point type within a third-generation Distributed Network Protocol (DNP3) device.

point type: The classification for entities having a common set of characteristics and attributes. Examples are binary inputs, analog inputs, counters, binary outputs, and analog outputs.

poll: A request for data from a master.

polling: An interrogate-reply scheme whereby a master schedules the transmission of requests for data to an outstation. Upon receipt of the request, the outstation returns the requested data in a response. The scheduled time of each poll and the specific data requested are a local matter.

primary station (when used in context of the Data Link Layer): The device (master or outstation) that initiates a message transaction between its Data Link Layer and that of a secondary device. The secondary, or non-initiating station, sometimes, but not always, depending on which function code is used by the primary, sends a response to complete the transaction.

private: Belonging to or restricted to an individual device or system and not generally known to other devices, systems, vendors, or persons. An example of a private application is a control loop implemented within a utility's outstation.

regenerated retry. The octets of this unsolicited response may contain some or all of the data from the previous unsolicited response and may also include updated data, new data, and changed internal indications (IIN) octets. The sequence number in the application control octet is incremented from the previously transmitted unsolicited response. Regeneration is permitted when the outstation does **not** receive confirmation to an unsolicited response.

remote mode: An operating condition whereby outputs may be controlled from a remotely located master. The outputs may also be operated locally if the system permits this.

report: To actually send to a master device a particular object variation. Used only in connection with outstation devices. An outstation may *parse* requests for a larger subset of objects than it actually *reports*.

report-by-exception: A schema whereby only changes are reported from an outstation. The data that remains constant is reported at infrequent intervals, via an integrity poll, as a means of assuring that the data in the master matches the data in the outstation. Report-by-exception is used for both polled and unsolicited responses.

request: An Application Layer message that asks an outstation to perform a specific action. A poll is only one type of request. There are other types of requests (e.g., actuate a control output and set the time).

responder: A station that responds or reacts to authentication challenges. May be either a master or an outstation.

response: An Application Layer message from an outstation that is returned to the master as the result of a request from the master.

secondary station (when used in context of the Data Link Layer): The device (master or outstation) that receives a request from a primary station.

segment: A packet of octets that is sized to fit into a Link Layer frame. Each segment contains a transport header and a portion of a fragment from the Application Layer.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Supervisory Control and Data Acquisition (SCADA): A generic term used to indicate monitoring and controlling devices that are physically located remotely from a centralized computer. Some form of communications are implied between the central location and the outlying devices.

subset: The Distributed Network Protocol (DNP) User's Group organization recognized that not all devices need to support all aspects of the DNP3 protocol, and that device complexity depends on its intented function within a system. To assure interoperability, various levels, called subset levels, are defined that specify behavior and parsing requirements. Each subset specifies a selection of DNP3 objects that a device shall be capable of

— Parsing when they appear in a message

— Processing if the respective data type is supported

— Formulating appropriate requests or responses with these objects

support: To be able to send or parse a particular set of third-generation Distributed Network Protocol (DNP3) objects, variations, function codes, and/or qualifiers.

transmitted unsolicited response. Any unsolicited response that has been sent from the outstation regardless of whether it was an original, an identical retry, or a regenerated retry response.

unsolicited response: An Application Layer message from an outstation to a master for which no explicit request was received. The request is implied by the act of a master enabling unsolicited reporting of various points within an outstation.

unsolicited response series. A sequence of unsolicited responses that begin with an original unsolicited response and terminate under conditions described later. An unsolicited response series may contain identical and regenerated retry messages.

3.2 Acronyms and abbreviations

ASCII	American Standard Code for Information Interchange
ASDU	Application Service Data Unit
ATM	Asynchronous Transfer Mode
BCD	binary-coded decimal
CA	certificate authority
CON	A bit in the Application Layer's control octet that specifies whether an Application Layer confirmation is required
CRC	cyclic redundancy check
CROB	control relay output block
DFC	Data Flow Control
DNP	Distributed Network Protocol

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3	The third generation of Distributed Network Protocol
EFCB	expected Frame Count bit
EPA	Enhanced Performance Architecture
EPRI	Electric Power Research Institute
FCB	Frame Count bit
FCV	Frame Count Valid
FIN	Final Data Link frame or final Application Layer fragment in a message
FIR	First Data Link frame or first Application Layer fragment in a message
FT#	Frame Type #
GPS	global positioning system
HMAC	Hash Message Authentication Code
IANA	Internet Assigned Numbers Authority
IEC	International Electrotechnical Commission
IED	intelligent electronic device
IERS	International Earth Rotation Service
IIN	internal indications
IP	Internet Protocol
ISO	International Organization for Standardization
LAN	local area network
LSB	Least Significant Byte
MSB	Most Significant Byte
NAT	network address translation
NIST	National Institute of Standards and Technology
NVRAM	non-volatile random access memory
OSI	Open System Interconnection
PCB	Pattern Control Block
PCM	Pattern Control Mask

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

RMS	root mean square
ROM	read-only memory
RTU	remote terminal unit
SCADA	Supervisory Control and Data Acquisition
SEQ	Sequence number that differentiates subsequent Data Link frames or Application Layer fragments. Sequence numbers associated with unsolicited responses are distinct from sequence numbers used for solicited responses
SHA	Secure Hash Algorithm
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UCA	Utility Communications Architecture
UDP	User Datagram Protocol
UNS	A bit in the Application Layer's control octet that specifies whether a fragment (response and confirmation) pertains to an unsolicited message. When this bit is set, the sequence number in the SEQ field refers to the unsolicited sequence number
UTC	Universal Coordinated Time
UTF	Unicode Transformation Format
UUID	Universally Unique Identifier
VT	virtual terminal
WAN	wide area network
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

3.3 Special terms

intelligent electronic device (IED): These are usually physically located close to sensors or actuators that are monitored or controlled.

outstation: A process that has data, variables, or information that another process wants to obtain or wants to set to a new value. May also refer to a device that contains an outstation process. An outstation can be a number of different devices/systems; some of these are intelligent electronic devices (IEDs) (remote terminal units, relays, meters, programmable logic controllers, distributed inputs/outputs, data concentrators/protocol converters, and substation human–machine interfaces), master stations, and regional operation center master stations.

remote terminal unit (RTU): Similar to an intelligent electronic device (IED), these are normally placed in the field close to sensors or actuators that are monitored or controlled. Often, RTUs are mounted outdoors and have demanding environmental specifications.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4. Application layer—part 1

4.1 Application layer preface

4.1.1 Layering

The Application Layer is the top layer in the EPA and OSI models. It interfaces with the DNP3 user's software and with the lower layers. Figure 1 shows where the layer fits within the layering stack. The Application Layer provides standardized functions, data formats, and procedures for the efficient transmission of data acquisition values, attributes, and control commands.

DNP3 user's software is the application program that makes a device unique, whether it is a master, IED, or a data concentrator. It makes use of the Application Layer's services to send messages to, and receive messages from, another DNP3 device.

Figure 1—Layering diagram

4.1.2 Introduction to points and point types

In DNP3, a *point* is a uniquely identifiable physical or logical entity. The term "point" applies to inputs like analogs, binaries, and counters and to outputs like analogs and binaries. A single analog input is an example of a point. It is associated with a specific measured signal or computed analog quantity. In addition to a value, an analog input may have a name, a scale factor, a deadband value for event detection, and other attributes.

A *point type* is a means of categorizing points having related characteristics, similar functionality, and relationship to physical hardware or logical space. Using the example from the previous paragraph, an analog input belongs to the analog input point type.

DNP3 models each point type as an independent array of points. Each point is uniquely identified by its index within the array. Figure 2 illustrates five basic point types within a DNP3 outstation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 2—Point type arrays

NOTE

A few DNP3 concepts associated with points are as follows:

— A point is identified by an index.

— A point usually has a static value.

— A point may generate events.

— Information about a point is reported using one or more objects.

Subsequent clauses define these in detail.

4.1.3 Introduction to indexes, groups, and variations

Figure 2 shows a DNP3 Outstation having five point types that it can transport. DNP3 can also transport files and other data forms. The transmitter shall suitably encode point information appearing within a message to enable the receiving device to parse and properly interpret this data. The index and group numbers identify a unique point, and the variation describes its data format.

4.1.3.1 Indexes

DNP3 uses index numbers to identify points having the same point type. Index numbers are equivalent to array element numbers. DNP3 indexes are zero-based; that is, the lowest element is always zero. Continuing with the analog input example, the first 10 analog input points are referenced using indexes 0 through 9.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.1.3.2 Groups

Groups provide a means of classifying the type or types of data within a message. Each group number shares a common point type and method of data generation or creation. Continuing with the example of an analog input point type, group numbers are used to report data as follows:

Group 30—current value of the point

Group 31—frozen value of the point

Group 32—change of current value event

Group 33—change of frozen value event

The group numbers listed are intended to illustrate the different data types that can be reported for the analog input point type; it is not required that all devices support these specific groups and data types—discussion of these issues occurs later.

Group numbers are also used for specifying the type of data needed for reporting time values, file controls, virtual terminal characters, and other information.

NOTE The term *object group* is often used in lieu of *group*. Group has many common meanings and is often intended as a collection of things. Object group is more suggestive of a set of point indexes associated with a specific point type whose data are conveyed by message objects. Objects are discussed in 4.1.4.

4.1.3.3 Variations

DNP3 offers a choice of encoding formats for many of the data types. The choices are known as variations. Every group number has an independent set of variations.

For example, group 30 has the following variations:

1—a 32-bit integer value with flag

2—a 16-bit integer value with flag

3—a 32-bit integer value

4—a 16-bit integer value

5—a 32-bit floating-point value with flag

6—a 64-bit floating-point value with flag

The flag refers to supplementary data that indicates conditions such as the source being online, the source device having restarted, or the value being outside of the measurement range. Refer to Annex A for specific details.

Variations are sometimes used in DNP3 to identify attribute values, counts of octets, and other formatting issues. Annex A provides details for all of the variations.

4.1.4 DNP3 objects

A DNP3 object is an encoded representation of data from a point, or other structure, that is formatted according to its group and variation number for transport in a message.

A message may contain multiple objects, each representing the value of a point at a given instant in time or a command to be issued to an output point.

The term *instance* is sometimes used to distinguish one object from another. For example, a message with 6 objects holds the current value of 6 analog input points, expressed as 32-bit integers. Here, the group number is 30, the variation is 3 and there are 6 instances of 32-bit analog input objects.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

This clause as well as Clause 5 through Clause 7 avoid referencing specific objects, unless needed for clarification. Object specifics appear in Annex A.

4.1.5 Static, event, and class data

4.1.5.1 Static

In DNP3, the term *static* refers to the point's current value, which is the most recently measured, computed, or obtained value. Thus, static binary input data refers to the present on or off condition of a bi-state point. Static analog input data contains the most recently obtained value of an analog input point. DNP3 allows the possibility to request some or all of the static data from an outstation device.

4.1.5.2 Events

DNP3 events are associated with something of significance happening. Examples are state changes, a measurement whose value crosses some threshold, an analog input changing by more than its deadband, a snapshot of data taken at a particular time, a transient phenomena, and some newly available information.

In many cases, the logic associated with points in an outstation can generate events, but other entities within an outstation can also cause creation of events. An outstation stores information about events in proprietary, vendor-specific structures.

Examples of the structured information stored for each event are as follows:

— Type of event (binary input, analog input, etc.).

— Value (on, off, 387, etc.).

— Point index.

— Time when event occurred.

— Class assignment.

— A representative object is in transmit buffer (true or false).

An event is generated or created when something occurs that requires storage of a new set of structured information about the event.

It is important to know the difference between **events** and the **DNP3 objects that report events**. Even though some DNP3 objects are referred to as "event objects," they are only a representation of the event, and not the actual structured information stored in the outstation. The stored event information may only be deleted from an outstation after an event description has been transmitted to the master (in a DNP3 object), and the outstation has received confirmation back from the master acknowledging that it received the event.

The three fundamental principles in DNP3 are as follows:

— The decision as to what constitutes an event is usually a local issue.

— It is not permitted to lose or discard the stored structured event information until a representation of the event has been transmitted to and acknowledged by the master.

— Duplicate reporting of an event should be avoided.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

The determination that a point has undergone a significant change may only be based on information from that point's current or past value. Decisions based on information about any other point, directly or indirectly, are explicitly forbidden.

There are DNP3 objects defined for reporting events with and without time stamps.

4.1.5.3 Classes

DNP3 uses the concept of *classes* to organize static data and events into several categories:

— Class 0: Static data (may be a subset of the outstation's total static data)

— Classes 1, 2, 3: Events

The points of most data types may be assigned to one of the four classes (Clause 5 details which data types may be assigned to which classes). If a point is assigned to Class 0, the point's present value shall be reported by the outstation in its response to a Class 0 poll, but the outstation shall not store or report any events for that point. If a point is assigned to one of the event classes (classes 1, 2, or 3), the outstation shall store and report events for that point, and the point's present value shall also be reported by the outstation in its response to a Class 0 poll. If a point is not assigned to any class, the outstation shall not include the point's present value in its response to a Class 0 poll, nor shall it store or report events for that point.

Note that if a point is not assigned to one of the four classes, the outstation shall still report its present value if the master performs a static poll of a specific data type. For example, if a master's request for binary input static data specifies all binary input points, or specifies the index of such a point (not assigned to one of the four classes), the outstation shall include that point's present value in its response.

If a point is not assigned to one of the event classes (classes 1, 2, or 3), the outstation shall not store any events for the point. Therefore, even if a master's request for binary input events specifies the index of such a point (not assigned to an event class), the outstation shall not have any stored events to report for that point.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 1—Function code table

	Point not assigned to any class	Point assigned to Class 0	Point assigned to Class 1	Point assigned to Class 2	Point assigned to Class 3
Present value included in response to static data-type poll?	Yes	Yes	Yes	Yes	Yes
Present value included in response to Class 0 poll?	No	Yes	Yes	Yes	Yes
Events stored and event data included in response to Class 1 poll?	No	No	Yes	No	No
Events stored and event data included in response to Class 2 poll?	No	No	No	Yes	No
Events stored and event data included in response to Class 3 poll?	No	No	No	No	Yes

DNP3 does not assign significance to the three event classes. One strategy is to assign the highest priority events to class 1 and the lowest priority events to class 3, but other strategies may work better in specific systems. Equipment manufacturers and persons configuring devices may decide which strategy is most appropriate for their application. Masters may request events from one or more of these classes.

Subclause 4.4.14 describes dynamically assigning events to event classes and contains informative material regarding event classes and statements about outstation configuration.

4.1.6 Outstation event buffering

This subclause is advisory to aid understanding. Its purpose is to discuss concepts related to retaining event information and to preventing loss or duplication of event data. It is not the intention to explicitly or implicitly specify an implementation design.

An event buffer is a structure in the outstation used to temporarily store event information. When an event is generated, software places information about the event into the buffer where it remains until relevant facts about the event have been transmitted to the master and their delivery confirmed. Upon receipt of the confirmation, event information is permanently removed from the buffer. Refer to 4.1.5.2 for additional discussion of event information.

DNP3 does not specify how many buffers an outstation maintains, one or multiple. Buffer structure, data organization, and contents are also not specified. These are considered vendor-specific design issues.

These examples are meant to illustrate event information. Depending on other implementation details and event types as well as other factors, a design may include more or less data for each event.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

As stated, event information is retained in the buffer until data has been sent to and confirmed by the master. This guarantees that events are not lost.

Removing event information from the buffer after transmission to, and confirmation back from, the master is sufficient to prevent reporting of the same event more than once when the device is used in a polled-only environment. Additional guards are required when an outstation is used in an unsolicited reporting environment.

Figure 3 illustrates buffering concepts. Please keep in mind that the diagram does not represent a required design.

Figure 3—Event buffering concepts

NOTE

> The descriptions in this specification sometimes use the term *event buffer* and at other times refer to *event buffers*. There is no significance to the use of the singular or plural form, and the reader is cautioned to think more about event buffering rather than about the number of buffers.

4.2 Message structure

Masters formulate and send request messages for an outstation to return data, carry out a command, or perform a special activity. Upon receipt, an outstation performs or initiates the requested action, generates an appropriate response message, and transmits it back to the master with the data, results, or special information.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

In certain systems, an outstation may spontaneously transmit an unsolicited response message to the master on the basis that the request is implied.

4.2.1 Application layer fragments

A *fragment* is a block of octets containing request or response information transported between a master and an outstation. Each fragment contains a function code specifying how the receiver shall process the fragment, zero or more data object headers and data-filled objects, and information for checking that the fragment is received in the proper order. Subclause 4.2.2 discusses the internal details of a fragment's structure.

DNP3 allows devices to limit the amount of memory reserved for sending and receiving messages. It achieves this by specifying the maximum length of each fragment and allowing response messages to be divided into one or multiple fragments. Small messages, requiring only a few octets, can fit into a single fragment, whereas larger messages may require multiple fragments.

Fragments are sometimes broken into smaller blocks when they are passed to the lower layers and then reassembled at the receiving end. This is illustrated in 0.2 of this standard.

Fragment rules are detailed in 4.3.

4.2.2 Application layer fragment structure

4.2.2.1 General fragment structure

Request and response fragments have similar, but slightly different, structuring (Figure 4).

Figure 4—Fragment structure

Each fragment begins with an application header that contains message control information. This is true for all fragments regardless of whether they appear in single or multiple fragment messages.

The application **response** header contains an additional field, called *internal indications*. This field does not appear in an application **request** header.

Often an application header alone is insufficient to convey complete information, and one or more sets of object headers and possibly DNP3 objects are included after the application header. These provide the additional information needed to form a complete message. Briefly, an object header specifies the type, format, and identification of the DNP3 objects that follow. A general description of object headers appears

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

in 4.2.2.7 of this document, and a complete list of DNP3 object descriptions and encoding are presented in Annex A of this standard.

4.2.2.2 Application request header

An application request header is used in requests from masters and has two fields (Figure 5). Each field is one octet in length. The fields are described in 4.2.2.4 and 4.2.2.5.

← Application Request Header →	
Application Control (1 octet)	Function Code (1 octet)

Figure 5—Application request header

4.2.2.3 Application response header

An application response header is used in responses from outstations and has three fields (Figure 6). The application control and function code fields are the same as in an application request header. The third field is two octets in length. The three fields are described in 4.2.2.4, 4.2.2.5, and 4.2.2.6.

← Application Response Header →		
Application Control (1 octet)	Function Code (1 octet)	Internal Indications (2 octets)

Figure 6—Application response header

4.2.2.4 Application control octet

The application control octet provides information needed to construct and reassemble multiple fragment messages and to indicate whether the receiver's Application Layer shall return an Application Layer confirmation message. It also provides information to assist in duplicate message detection.

The bits in an application control octet form five fields as in Figure 7.

Bit # →	7	6	5	4	3	2	1	0
Fields →	FIR	FIN	CON	UNS	SEQ			

Figure 7—Application control octet fields

4.2.2.4.1 FIR field

The FIR field is a single bit, which when set indicates that this is the first fragment of a message.

FIR = 0 indicates this is **not** the first fragment of a message.

FIR = 1 indicates this **is** the first fragment of a message.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.2.2.4.2 FIN field

The FIN field is a single bit, which when set indicates that this is the final fragment of a message.

FIN = 0 indicates that **more fragments** follow.

FIN = 1 indicates this **is** the final fragment of a message.

4.2.2.4.3 CON field

The CON field is a single bit, which when set indicates that the receiver's Application Layer shall return an application confirmation message.

CON = 0 indicates an Application Layer confirmation message shall **not** be returned.

CON = 1 indicates an Application Layer confirmation message **shall be** returned.

An Application Layer confirmation message is a very brief message that is used to verify that a complete fragment arrived at its destination.

— An outstation shall set the CON bit in response messages containing event data. Confirmation back from the master indicates that the event information arrived at the master was properly understood and processed and that the outstation can safely discard the events whose data appeared in the response.

— The CON bit shall be set for each fragment of a multi-fragment response message with the exception of the last fragment. Setting the CON bit in the last fragment is optional; it may be set or cleared if desired unless other reasons require it to be set. Receipt of the confirmation signifies that the device may safely transmit the next fragment.

— An outstation shall set the CON bit in unsolicited response messages. Receipt of the confirmation indicates that the master received the message and that the outstation can advance its state logic.

NOTE

> Outstations that have no events available when the master requests events are encouraged to clear the CON bit in the response (not request confirmation) unless there is another reason for setting that bit. Confirming the receipt of no events does not serve a useful purpose.

— Masters shall never request Application Layer confirmation; i.e., the CON bit shall never be set in a request message. This practice is obsolete.

4.2.2.4.4 UNS field

The UNS field is a single bit, which when set indicates the message contains an unsolicited response or a confirmation of an unsolicited response.

UNS = 0 indicates the sequence number is associated with a master request or a solicited response message.

UNS = 1 indicates the sequence number is associated with an unsolicited response message.

Outstations set this bit in fragments that contain an unsolicited response. Masters set this bit in Application Layer confirmation fragments that confirm the receipt of an unsolicited response.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Outstations clear this bit in solicited response fragments. Masters clear this bit in request fragments and in Application Layer confirmation fragments that confirm the receipt of a solicited response.

4.2.2.4.5 SEQ field

The SEQ field is 4 bits wide. It is used to verify that fragments are received in the correct order and to detect duplicated fragments. The SEQ field has a range of 0 to 15. After the sequence number of 15, the next value is 0. The number increments by one count, modulo 16, for non-repeated

— Master request fragments.

— Subsequent fragments after the first in a multi-fragment message.

— Unsolicited response fragments.

A device that supports both solicited and unsolicited responses shall maintain **independent** sequence numbers for every other device with which it communicates:

— One is used for solicited requests, responses, and confirmations.

— The other is used for unsolicited responses and confirmations.

No relationship exists between the two sequence numbers.

4.2.2.5 Function code octet

The function code octet identifies the purpose of the message. Request messages from masters use function codes in the range of 1 to 128, and response messages from outstations use function codes with values ranging from 129 to 255 as shown in Table 2. Application Layer confirmations use the CONFIRM function code.

Table 2 gives a brief description of each function code. Sublause 4.4 of this document provides detailed procedures for each function code. The subclause numbers referenced in the right-hand column indicate where the respective details are located.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 2—Function code table

Message type	Code	Name	Brief description
Confirmation	0 0x00	CONFIRM	Confirm Function Code: Master sends this to an outstation to confirm the receipt of an Application Layer fragment. Reference: 4.4.1
Request	1 0x01	READ	Read Function Code: Outstation shall return the data specified by the objects in the request. Reference: 4.4.2
Request	2 0x02	WRITE	Write Function Code: Outstation shall store the data specified by the objects in the request. Reference: 4.4.3
Request	3 0x03	SELECT	Select Function Code: Outstation shall select (or arm) the output points specified by the objects in the request in preparation for a subsequent operate command. The outstation shall not activate the outputs until a request with a matching Operate function code is received. Reference: 4.4.4
Request	4 0x04	OPERATE	Operate Function Code: Outstation shall activate the output points selected (or armed) by a previous select function code command. Reference: 4.4.4
Request	5 0x05	DIRECT_OPERATE	Direct Operate Function Code: Outstation shall immediately actuate the output points specified by the objects in the request. A prior matching select command is not required. Reference: 4.4.5
Request	6 0x06	DIRECT_OPERATE_NR	Direct Operate—No Response Function Code: Same as function code 5 but outstation shall not send a response. Reference: 4.4.5
Request	7 0x07	IMMED_FREEZE	Immediate Freeze Function Code: Outstation shall copy the point data values specified by the objects in the request to a separate freeze (or holding) buffer (or register). Reference: 4.4.6

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Message type	Code	Name	Brief description
Request	8 0x08	IMMED_FREEZE_NR	Immediate Freeze—No Response Function Code: Same as function code 7 but outstation shall not send a response. Reference: 4.4.6
Request	9 0x09	FREEZE_CLEAR	Freeze and Clear Function Code: Outstation shall copy the point data values specified by the objects in the request into a separate freeze (or holding) buffer (or register). After the copy operation, clear the point data values to zero. Reference: 4.4.7
Request	10 0x0A	FREEZE_CLEAR_NR	Freeze and Clear—No Response Function Code: Same as function code 9 but outstation shall not send a response. Reference: 4.4.7
Request	11 0x0B	FREEZE_AT_TIME	Freeze at Time Function Code: Outstation shall copy the point data values specified by the objects in the request to a separate freeze (or holding) buffer (or register) at the time and/or time intervals specified in a special time data information object. Reference: 4.4.8
Request	12 0x0C	FREEZE_AT_TIME_NR	Freeze at Time—No Response Function Code: Same as function code 11 but outstation shall not send a response. Reference: 4.4.8
Request	13 0x0D	COLD_RESTART	Cold Restart Function Code: Outstation shall perform a complete reset of all hardware and software in the device. Reference: 4.4.9
Request	14 0x0E	WARM_RESTART	Warm Restart Function Code: Outstation shall reset only portions of the device. Reference: 4.4.9
Request	15 0x0F	INITIALIZE_DATA	Initialize Data Function Code: Obsolete—Do not use for new designs. Reference: 4.4.10
Request	16 0x10	INITIALIZE_APPL	Initialize Application Function Code: Outstation shall place the applications specified by the objects in the request into the ready to run state. Reference: 4.4.11

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Message type	Code	Name	Brief description
Request	17 0x11	START_APPL	Start Application Function Code: Outstation shall start running the applications specified by the objects in the request. Reference: 4.4.11
Request	18 0x12	STOP_APPL	Stop Application Function Code: Outstation shall stop running the applications specified by the objects in the request. Reference: 4.4.11
Request	19 0x13	SAVE_CONFIG	Save Configuration Function Code: Outstation shall store into non-volatile memory the contents of a configuration file located in volatile memory. This code is deprecated—Do not use for new designs. Reference: 4.4.12
Request	20 0x14	ENABLE_UNSOLICITED	Enable Unsolicited Responses Function Code: Enables outstation to initiate unsolicited responses from points specified by the objects in the request. Reference: 4.4.13
Request	21 0x15	DISABLE_UNSOLICITED	Disable Unsolicited Responses Function Code: Prevents outstation from initiating unsolicited responses from points specified by the objects in the request. Reference: 4.4.13
Request	22 0x16	ASSIGN_CLASS	Assign Class Function Code: Outstation shall assign the events generated by the points specified by the objects in the request to one of the classes. Reference: 4.4.14
Request	23 0x17	DELAY_MEASURE	Delay Measurement Function Code: Outstation shall report the time it takes to process and initiate the transmission of its response. This allows the master to compute the propagation delay in the communications channel. Used for non-LAN time synchronization. Reference: 4.4.15
Request	24 0x18	RECORD_CURRENT_TIME	Record Current Time Function Code: Outstation shall save the time when the last octet of this message is received. Used for LAN time synchronization. Reference: 4.4.16

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Message type	Code	Name	Brief description
Request	25 0x19	OPEN_FILE	Open File Function Code: Outstation shall open a file. Reference: 4.4.17
Request	26 0x1A	CLOSE_FILE	Close File Function Code: Outstation shall close a file. Reference: 4.4.17
Request	27 0x1B	DELETE_FILE	Delete File Function Code: Outstation shall delete a file. Reference: 4.4.17
Request	28 0x1C	GET_FILE_INFO	Get File Information Function Code: Outstation shall retrieve information about a file. Reference: 4.4.18
Request	29 0x1D	AUTHENTICATE_FILE	Authenticate File Function Code: Outstation shall return a file authentication key. Reference: 4.4.19
Request	30 0x1E	ABORT_FILE	Abort File Function Code: Outstation shall abort a file transfer operation. Reference: 4.4.17
Request	31 0x1F	ACTIVATE_CONFIG	Activate Configuration Function Code: Outstation shall use the configuration specified by the objects in the request. Reference: 4.4.20
Request	32 0x20	AUTHENTICATE_REQ	Request to authenticate a response from the outstation. Reference: 4.4.21
Request	33 0x21	AUTHENTICATE_ERR	Notice of an authentication error. Reference: 4.4.22
	32 0x20 to 128 0x80		Reserved.
Response	129 0x81	RESPONSE	Solicited Response Function Code: Master shall interpret this fragment as an Application Layer response to an Application Layer request sent by the master. Reference: 4.4.20

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Message type	Code	Name	Brief description
Response	130 0x82	UNSOLICITED_RESPONSE	Unsolicited Response Function Code: Master shall interpret this fragment as an unsolicited response that was not prompted by an explicit request. Reference: 4.4.24
Response	131 0x83	AUTHENTICATE_RESP	Authentication responses sent from outstation Reference: 4.4.25
Response	132 0x84 to 255 0xFF		Reserved.

4.2.2.6 Internal indications

The internal indications field appears in application response headers immediately following the function code octet. This field has two octets. The bits in these two octets indicate certain states and error conditions within the outstation (Figure 8).

Figure 8—Internal indications octets

Each octet has 8 bit fields numbered 0 through 7 where bit 0 is the least significant bit.

NOTE

This document uses a code to reference or specify a particular bit:

IINx.b

Where x is a 1 for the first octet and x is a 2 for the second.

b identifies the bit number.

Thus, IIN2.0 refers to the first bit in the second octet.

To simplify referencing the IIN bits in code and other documentation, each IIN bit has a name as shown in the Table 3. A brief description of each bit is given in the table. Detailed descriptions are provided in 4.5 of this document. The subclause numbers referenced in the right-hand column indicate where the respective details are located.

Readers are cautioned to read the detailed descriptions in 4.5 because the brief descriptions provided in Table 2 do not adequately express the complete requirements and interpretation of the IIN bits.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 3—IIN bits

Bit	Name	Brief description
IIN1.0	ALL_STATIONS	An all-stations message was received. Reference 4.5.1
IIN1.1	CLASS_1_EVENTS	The outstation has unreported class 1 events. Reference 4.5.2
IIN1.2	CLASS_2_EVENTS	The outstation has unreported class 2 events. Reference 4.5.3
IIN1.3	CLASS_3_EVENTS	The outstation has unreported class 3 events. Reference 4.5.4
IIN1.4	NEED_TIME	Time synchronization is required. Reference 4.5.5
IIN1.5	LOCAL_CONTROL	One or more of the outstation's points are in local control mode. Reference 4.5.6
IIN1.6	DEVICE_TROUBLE	An abnormal, device-specific condition exists in the outstation. Reference 4.5.7
IIN1.7	DEVICE_RESTART	The outstation restarted. Reference 4.5.8
IIN2.0	NO_FUNC_CODE_SUPPORT	The outstation does not support this function code. Reference 4.5.9
IIN2.1	OBJECT_UNKNOWN	Outstation does not support requested operation for objects in the request. Reference 4.5.10
IIN2.2	PARAMETER_ERROR	A parameter error was detected. Reference 4.5.11
IIN2.3	EVENT_BUFFER_OVERFLOW	An event buffer overflow condition exists in the outstation, and at least one unconfirmed event was lost. Reference 4.5.12
IIN2.4	ALREADY_EXECUTING	The operation requested is already executing. Support is optional. Reference 4.5.13
IIN2.5	CONFIG_CORRUPT	The outstation detected corrupt configuration. Support is optional. Reference 4.5.14
IIN2.6	RESERVED_2	Reserved for future use. Always set to 0. Reference 4.5.15
IIN2.7	RESERVED_1	Reserved for future use. Always set to 0. Reference 4.5.16

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.2.2.7 Object headers

Subclause 4.2.2.1 shows optional sets of object headers and DNP3 objects following the application request and response headers. That clause indicated those object headers and DNP3 objects provide additional information when the application header alone cannot convey the complete message.

To explain why object headers are needed, consider a master that wants to read 20 analog values from a remote terminal unit (RTU) device. In formulating the request message, the master uses the function code READ, which specifies the action. The object header (or headers) in the request specifies

— The analog input point type data is wanted.

— The integer or floating-point format the outstation should use when sending its data.

— Which particular 20 input point indexes.

DNP3 objects are **not** included in the request, only an object header (or headers), because the master is not sending values; it only sends enough information for the outstation to know which values and format are desired. The response message uses function code RESPONSE and contains the same or a similar object header (or headers) followed by the DNP3 objects. Each object consists of a single analog value from a single point index.

This subclause elaborates on the construction of object headers. Clause 11 specifies the details of individual DNP3-supported objects.

Object headers consist of mandatory object type and qualifier fields and a range field that is contingent on the code in the qualifier field. The object type field consists of a group octet followed by a variation octet (Figure 9).

Figure 9—Object header fields

4.2.2.7.1 Object group

The object group specifies what data type or values are included in a master request or in an outstation response. A complete collection of object groups appear in Annex A. A few examples are as follows:

— Analog input, current value.

— Binary input, event value.

— Frozen counter, current value.

— Control relay output block (CROB) value.

— Time value.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.2.2.7.1.1 Object group 60

Object group 60 is a special group number assigned for requesting data from one of the class categories. Only a master request may include this object group in a message. Depending on the associated object variation (explained in 4.2.2.7.2), the master includes this object in a message when it wants an outstation to return all or some of its class 0 static data or class 1, 2, or 3 event data.

The response does not use object group 60 because it is not specific enough to identify the data in the response. Instead, the outstation's response shall contain specific group identifiers for each of the different data types in the message. As an example of an outstation's response resulting from a request for data from group 60, variation 1, the device sends object headers with group 1, binary input data, and group 30, analog input data; each object header is followed by the respective data-filled objects.

4.2.2.7.2 Object variation

The object variation specifies the data format of DNP3 objects. For example, variations exist for reporting the current value of analog inputs as 16-bit integers, 32-bit integers, short floating-point quantities, and long floating-point quantities; the integer types may include an additional flag octet. Each alternative format has a unique variation number.

The object group and object variation together uniquely specify the structure of a DNP3 object and the type of data that it references.

4.2.2.7.2.1 Variation 0

Variation 0 has special meaning, and only master requests may use it. Variation 0 indicates that the master does not have a preferred format for the outstation to use in its response—the outstation may determine which object variations to return. For example, when the master issues a request for analog inputs, group 30, variation 0, to outstation ABC, the response returned might contain group 30 objects formatted per variation 2, 16-bit value with flag (3 octets). And if the same request were issued to outstation DEF, it might respond with group 30 objects formatted per variation 1, 32-bit value with flag (5 octets).

NOTE

> It is suggested that outstation designers provide configuration for setting which variations are reported in responses when the master request specifies variation 0.

It is permissible for individual indexes of an object group to have different variations in a response when the master request specifies variation 0. It is not necessary for the outstation to transmit the identical variation for all of the objects having a common group number.

Masters that use variation 0 shall be prepared to accept, as a minimum, all of the specific variations declared for the DNP3 implementation level that they support. See Clause 14 for more details.

NOTE— From this point forward, this document uses the shorthand notation "**g**NN**v**MM" for referencing specific objects by group and variation number. gNN stands for object group NN, and vMM stands for object variation MM. As an example, g12v1 means group 12, variation 1.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.2.2.7.3 Qualifier and range fields

These two fields are discussed together because the structure and contents of the range field is dependent on the qualifier octet's value (Figure 10).

Bit #	7	6	5	4	3	2	1	0
Fields	Res	Object Prefix Code			Range Specifier Code			

Figure 10—Qualifier octet fields

The qualifier octet is divided into three fields.

4.2.2.7.3.1 Res field

This field is reserved for future use. It is one bit wide and shall be set to 0 in current implementations.

4.2.2.7.3.2 Object prefix code

The object prefix code is three bits wide. It specifies what, if any, prefix value appears before each of the DNP3 objects that follow the object header. Prefixes are either an index number or an object size.

Object Header	1st Prefix	1st DNP3 Object	2nd Prefix	2nd DNP3 Object	• • •	Nth Prefix	Nth DNP3 Object

Figure 11—Objects prefixed

Figure 11 illustrates objects preceded by a prefix.

It is sometimes necessary to place a prefix before each of the objects to associate the object data with an index in the outstation. (Indexes were discussed in 4.1.3.1.) An example is a list of class 1 events. The master cannot know ahead of time which input points changed state; therefore, when sending event data, it is normal for the outstation to prefix the object data with an index number. In this case, the prefix provides the information for the master to correctly correlate the new data value with a point in the outstation.

At other times, when the object group and variation do not specifically imply the object size, it is necessary to place a prefix before each object giving its size.

Table 4 enumerates the DNP3 object prefix codes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 4—Object prefix codes

Code (Hex)	Description	Size of object prefix
0	Objects are packed without an index prefix.	—
1	Objects are prefixed with an index.	1-octet
2	Objects are prefixed with an index.	2-octet
3	Objects are prefixed with an index.	4-octet
4	Objects are prefixed with an object size.	1-octet
5	Objects are prefixed with an object size.	2-octet
6	Objects are prefixed with an object size.	4-octet
7	Reserved for future use.	—

In some cases, such as reading a list of non-sequential binary input points, the master needs to convey a list of point indexes to the outstation; there are no real data-filled objects for the master to transmit. In this circumstance, the master uses object prefix code 1, 2, or 3 and sends "null objects" (having zero octets) prefixed by a point index. This appears in the request as an object header followed by a list of point indexes. See Figure 12.

Figure 12—Index list

In other situations, data is from contiguous point indexes. The range field specifies a start index and stop index, and a prefix is not necessary because the index correlates to the position of the data within the response. Bandwidth is saved by not including redundant index values. Code 0 is used to specify that no prefix codes appear before the objects.

4.2.2.7.3.3 Range specifier codes

The range specifier code indicates whether a range field is used and, if so, what it contains and how large it is.

Table 5 enumerates the range specifier codes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 5—Range specifier codes

Code (Hex)	Description	Number of octets in range field
0	Range field contains 1-octet start and stop indexes.	2
1	Range field contains 2-octet start and stop indexes.	4
2	Range field contains 4-octet start and stop indexes.	8
3	Range field contains 1-octet start and stop virtual addresses.	2
4	Range field contains 2-octet start and stop virtual addresses.	4
5	Range field contains 4-octet start and stop virtual addresses.	8
6	No range field is used. This implies all values.	0
7	Range field contains 1-octet count of objects.	1
8	Range field contains 2-octet count of objects.	2
9	Range field contains 4-octet count of objects.	4
A	Reserved for future use.	—
B	Variable format qualifier, range field contains 1-octet count of objects.	1
C	Reserved for future use.	—
D	Reserved for future use.	—
E	Reserved for future use.	—
F	Reserved for future use.	—

Range specifier codes 0 through 2

Used when the DNP3 objects are packed contiguously in index order. The index of the first object shall be the start index. The index of each succeeding object shall be one greater than the preceding object's index. The index of the last object is the stop index. When the start and stop index values are identical, the range specifies only a single object.

Range specifier codes 3 through 5

Used for specifying contiguous address locations in a vendor-specific virtual memory space. These codes are not widely used, but when a vendor implements them, they are most often used with 8-bit unsigned integer objects. See group 102 in the *Object Library*. The address of the first object shall be the start address. The address of each succeeding object shall be one greater than the preceding object's address. The address of the last object is the stop address. When the start and stop address values are identical, the range specifies only a single object.

Range specifier code 6

Indicates the master desires values from all of the points specified by the object group. There is no range field.

Range specifier codes 7 through 9

Indicate the range field contains a count of data objects or a count of object indexes.

An outstation may respond with fewer objects if a master's request uses qualifier codes 0x07, 0x08, or 0x09 and there is a practical reason for sending a lesser number. Outstations shall interpret these qualifiers as meaning "as many as, but not more than the count appearing in the range field." An outstation shall send at least one object if it has data of the type requested.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Range specifier code 0xB

Indicates the data has a variable length format that cannot be predefined for all objects of this group and variation.

4.2.2.7.3.4 Valid qualifier codes

Table 6 shows the codes that are permitted if both transmitting and receiving devices support them. Shaded cells show codes reserved for future use. The qualifier codes that specify a four-octet index (0x02, 0x37, 0x38, and 0x39) should never be used unless a device has more than 65 536 points. Codes specifying four-octet sizes (0x6B) or counts (0x09, 0x19, 0x29, and 0x39) should be avoided because, in practice, object sizes and counts can be described using the one or two-octet codes.

Table 6—Valid qualifier codes

Valid Qualifier Codes (Hexadecimal)																
Range Specifier → / Object Prefix	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	04	05	06	07	08	09						
1								17	18	19						
2								27	28	29						
3								37	38	39						
4												4B				
5												5B				
6												6B				
7																

4.2.2.7.3.5 Preferred qualifier codes

Only a few of the valid qualifier codes are recommended. The matrix in Table 7 shows which combinations of object prefix codes and range specifier codes are preferred for the sake of interoperability between devices.

Table 7—Preferred qualifier codes

Preferred Qualifier Codes (Hexadecimal)																
Range Specifier → / Object Prefix	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01					06	07	08							
1								17								
2									28							
3																
4																
5												5B				
6																
7																

The subset definitions only use the preferred qualifier codes in specifying which codes master and outstation devices shall support. Table 8 lists the qualifier codes that were chosen for the subsets and examples of their intended use.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 8—Qualifier codes used by subset requirements

Qualifier code (Hex)	Use in a master request	Use in an outstation response
00, 01	A single point or range of static points.	Static objects.
06	All points.	Not permitted in a response.
07, 08	A limited quantity of events. A single quantity having no index (e.g., Time and Date).	A single quantity having no index (e.g., Time and Date).
17, 28	For controls or other functions, such as reading, that request multiple objects where the indexes are non-sequential or not consecutive.	Event objects (usually one or more unrelated points).
5B	To transmit objects whose size may be unknown to the receiver (e.g., File Open).	To transmit objects whose size may be unknown to the receiver (e.g., File data).

4.2.2.7.4 Qualifier examples

EX 2	Qualifier 0x00 example. This qualifier is used in requests and responses when objects from contiguous indexes are desired or being transported.

Range Field In Header: 1-octet start–stop indexes

I	I + 2
Start	Stop

Data Objects: Three objects are shown having consecutive indexes beginning at index I

Value	Value	Value
LSB ⋮ • • • ⋮ MSB	LSB ⋮ • • • ⋮ MSB	LSB • • • MSB
Object (index I)	Object (index I + 1)	Object (index I + 2)

EX 3	Qualifier 0x01 example. This qualifier is used in requests and responses when objects from contiguous indexes are desired or being transported.

Range Field In Header: 2-octet start–stop indexes

I	I + 3
LSB ⋮ MSB	LSB ⋮ MSB
Start	Stop

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Data Objects: Four objects are shown having consecutive indexes beginning at index I

Value	Value	Value	Value
LSB ⋮ • • • ⋮ MSB	LSB ⋮ • • • ⋮ MSB	LSB ⋮ • • • ⋮ MSB	LSB ⋮ • • • ⋮ MSB
Object (index I)	Object (index I + 1)	Object (index I + 2)	Object (index I + 3)

EX 4	Qualifier 0x06 example. This qualifier is only used in requests to indicate all points defined by the group and variation in the object header.

Range Field In Header: No range field—implies all objects

Data Objects: There are no data objects.

EX 5	Qualifier 0x07 example. This qualifier is used in requests to indicate a limited quantity of events or a single quantity that does not have an associated index (e.g., Time and Date). It is used in a response for a single value that is not associated with an index (e.g., Time and Date).

Range Field In Header: 1-octet count of objects

0x01
Count

Data Objects: When requesting events, there are no objects. There is only one object in a request to write a time, and responses to time and time delay requests contain only a single object.

Value
LSB ⋮ • • • ⋮ MSB
Object

EX 6	Qualifier 0x08 example. This qualifier is used in requests to indicate a limited quantity of events or a single quantity that does not have an associated index (e.g., Time and Date). It is used in a response for a single value that is not associated with an index (e.g., Time and Date).

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Range Field In Header: 2-octet count of objects

0x0001
LSB ⋮ MSB
Count

Data Objects: When requesting events, there are no objects. There is only one object in a request to write a time, and responses to time and time delay requests contain only a single object.

Value
LSB ⋮ • • • ⋮ MSB
Object

EX 7	Qualifier 0x17 example. This qualifier is used in request for controls for one or more non-contiguous points. It is used in responses to report events or other values where indexes are not contiguous.

Range Field In Header: 1-octet count of objects

0x03
Count

Data Objects: Example shows three objects, each with a 1-octet index prefix.

M	Value	k	Value	j	Value
	LSB ⋮ • • • ⋮ MSB		LSB ⋮ • • • ⋮ MSB		LSB ⋮ • • • ⋮ MSB
Index	Object (index m)	Index	Object (index k)	Index	Object (index j)

EX 8	Qualifier 0x28 example. This qualifier is used in request for controls for one or more non-contiguous points. It is used in responses to report events or other values where indexes are not contiguous.

Range Field In Header: 2-octet count of objects

0x0003
LSB ⋮ MSB
Count

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Data Objects: Example shows three objects, each with a 2-octet index prefix.

m	Value	k	Value	j	Value
LSB ⋮ MSB	LSB ⋮ • • • ⋮ MSB	LSB ⋮ MSB	LSB ⋮ • • • ⋮ MSB	LSB ⋮ MSB	LSB ⋮ • • • ⋮ MSB
Index	Object (index m)	Index	Object (index k)	Index	Object (index j)

EX 9	Qualifier 0x5B example. This qualifier is used in requests and responses to transmit data whose size may be unknown to the receiver.

Range Field In Header: 1-octet count of objects

0x02
Count

Data Objects: Example shows two objects, each with a 2-octet size prefix

LSB ⋮ MSB	LSB ⋮ • • • ⋮ MSB	LSB ⋮ MSB	LSB ⋮ • • • ⋮ MSB
Size	First Object	Size	Second Object

4.3 Fragment rules

NOTE—The FIR, FIN, CON, and UNS bits and the SEQ values in the rules refer to contents of these fields in the application control octet.

— **Rule 1:** DNP3 devices that are able to set their maximum transmit fragment size larger than 2048 octets shall provide the ability to limit the size (via configuration) to a maximum of 2048 octets.

— **Rule 2:** All devices shall accept fragments as small as 2 octets.

— **Rule 3:** Outstations shall be prepared to receive fragment sizes of at least 249 octets, and masters shall be prepared to receive fragment sizes of at least 2048 octets.

— **Rule 4:** Master devices shall only send requests that fit within a single fragment.

NOTE

A master should be aware of the maximum fragment size that each of its outstations is able to receive. Simple devices may not have, nor require much memory, and consequently may limit the fragment size to as low as 249.

NOTE

Whenever a master has more data to send than fits within a single fragment (such as a file transfer, setting analog deadbands, etc.), it is the master's responsibility to formulate smaller requests and send each separately.

— **Rule 5:** Master devices shall accept multiple fragment responses.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— **Rule 6:** An outstation device shall be able to return all of its event and static data together, within a single response. If necessary, it shall use multiple fragments to convey the entire response.

— **Rule 7:** Each fragment shall be individually and completely parsable. A fragment shall only contain complete DNP3 objects and not portions of an object; that is, objects may not be divided across two or more fragments.

— **Rule 8:** The FIR bit shall be set in the fragment that begins a message.

— **Rule 9:** The FIN bit shall be set in the fragment that ends a message.

— **Rule 10:** A message may consist of a single fragment having both the FIR and FIN bits set.

— **Rule 11:** Masters shall never request Application Layer confirmation; i.e., they shall not set the CON bit in request messages. This practice is obsolete.

— **Rule 12:** Outstations that receive a properly formatted fragment with the CON bit set shall immediately respond with an Application Layer confirm message (for backward compatibility).

— **Rule 13:** Masters that receive a properly formatted fragment with the CON bit set shall respond with an Application Layer confirm message before sending any other request message to that outstation.

— **Rule 14:** For each new, non-retry request fragment it sends, a master shall increment the SEQ number by one count (modulo 16) from its previous request fragment.

— **Rule 15:** The first fragment of a **solicited** (polled) **response** message shall have the same SEQ number as the SEQ number in the request fragment. If the response requires multiple fragments, each subsequent fragment shall use a SEQ number incremented by one count (modulo 16) from the previous.

— **Rule 16:** A master may send a retry request message if it uses the same SEQ number and all the other octets in the retry message match the original request message. There are exceptions where retries are prohibited, namely, if the request includes

1) Function codes DIRECT_OPERATE or DIRECT_OPERATE_NR—see 4.4.5 for more information.
2) Function codes DELAY_MEASURE or RECORD_CURRENT_TIME—see Clause 10.
3) Function code WRITE with an Absolute Time object, g50v1, or with a Last Recorded Time object, g50v3—see Clause 10.

— **Rule 17:** An outstation shall not retry sending **solicited response** messages. (This requirement does not affect the Data Link Layer, which may retry Data Link frames if appropriate.) An outstation may retry sending **unsolicited response** messages—refer to the rules in 4.6.6.

— **Rule 18:** A fragment containing a CONFIRM function code shall use the same SEQ number and UNS bit state as are in the fragment being confirmed.

— **Rule 19:** Outstations shall ignore the SEQ number in broadcast request messages.

— **Rule 20:** The SEQ numbers sent by an outstation in unsolicited responses are distinct from, and have no relationship to, the SEQ numbers it sends in solicited responses.

— **Rule 21:** An outstation that sends unsolicited responses may choose any SEQ number for its first message after a restart. The master shall accept these messages.[9]

[9] There are additional rules for devices that send unsolicited messages in 4.6.6, but briefly, an outstation's initial unsolicited response after a restart shall be a null message that does not include any DNP3 objects and has the IIN1.7 [DEVICE_RESTART] bit set.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— **Rule 22:** An outstation that restarts shall ignore the SEQ number in the first request it receives from a master after the restart but otherwise shall execute the request. Thereafter, the outstation should examine the SEQ numbers in the received requests.

— **Rule 23:** An outstation shall request an Application Layer confirmation when it sends a fragment containing event objects. Receipt of the confirmation message from the master lets the outstation know that the event information arrived at the master and, therefore, that the outstation may discard corresponding event data from its event buffers.

NOTE

> It is important to differentiate **event data** from **event objects**. An event is the occurrence of something significant happening, and **event data** is the information that is retained in the outstation regarding an event. An outstation uses **event data** to create **event objects** that are transmitted in response messages to the master. Only after confirmation may the outstation discard the corresponding **event data**.

— **Rule 24:** An Application Layer confirmation is required for each fragment of a multi-fragment message except the last fragment. Application Layer confirmation is optional for the last fragment of a multi-fragment message unless there is another reason why confirmation is mandatory, such as the last fragment contains events. Receipt of the confirmation signifies to the outstation that it may send the next fragment. It also informs the outstation that it can safely discard any event data in the confirmed fragment. Outstations shall not send the next fragment of a multi-fragment response until the master confirms the outstation's previously transmitted fragment.

— **Rule 25:** Application Layer confirmation is required to acknowledge receipt of unsolicited response messages. An outstation shall not discard any information or make any assumptions about what the master received if it does not receive a confirm message from the master.

— **Rule 26:** Outstations are sometimes required to request Application Layer confirmation after a master sends an all-stations (broadcast) request. The particular all-stations address that the master uses in the message determines the need for confirmation. See 4.5.1 for details.

4.4 Detailed function code procedures

This clause contains implementation information, rules, and recommendations for each of the function codes.

4.4.1 Function code 0

Confirm

A master sends a message with this function code to confirm receipt of a response fragment.

An Application Layer confirmation message is a very brief message. It consists of the application control octet and the function code octet. There are no object headers or DNP3 object octets.

Masters only send confirmation messages when outstations request them.

An outstation requests the master to confirm the receipt of a fragment by setting the CON bit in the application control header. See 4.2.2.4.3.

A master is obligated to confirm messages that it receives with the CON bit set in the application control header.

Refer to the rules in 4.3 for when outstations should request a confirmation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.4.2 Function code 1 (0x01)

Read

The READ function code is the basic code used by a master to request data from an outstation.

The object headers in the request specify which data the master desires and/or how many objects and sometimes what format to use in the response. A request message may contain more than one object header, thereby effectively combining several requests into a single message.

Responses contain zero, one, or more object headers; each object header is followed by its respective DNP3 objects.

The general formats of ***read*** request and response messages are shown in the following discussion. To keep the diagrams simple, a single fragment response is assumed.

AC is the Application Control octet.

FC is the Function Code octet.

$IIN_{1,2}$ represents the Internal Indication octets.

OH_x is the xth Object Header (shown shaded).

DIO_{xy} is the yth Data Information Object associated with the xth object header.

►►► Request Message

AC	FC	OH_0	OH_1	• • •	OH_n

◄◄◄ Response Message (beginning)

AC	FC	$IIN_{1,2}$	OH_0	DIO_{00}	DIO_{01}	• • •	DIO_{0i}	OH_1	DIO_{10}	DIO_{11}

◄◄◄ Continuation of Response Message

• • •	DIO_{1j}	• • •	OH_n	DIO_{n0}	DIO_{n1}	• • •	DIO_{nk}

4.4.2.1 Read rules

— **Rule 1:** If an outstation is waiting for an Application Layer confirmation to a previously transmitted **solicited response** message (because the master requested a **read**), and instead it receives a new request of any kind, the outstation shall

1) Assume the confirmation is not forthcoming.
2) Retain the event data.
3) Cancel the previous transaction.
4) Process and respond to the new request as if there were no Application Layer confirmation outstanding.

This ensures that control operations and other vital functions receive priority over pending read operations. It also guarantees that the master retains control over polling sequences.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— **Rule 2:** An outstation shall delay formulation and transmission of a response when it receives a request having a READ function code while awaiting an Application Layer confirmation to a previously transmitted **unsolicited response** message. This requirement is necessary to prevent the outstation from prematurely removing events from its buffers. See 4.6.6 and 4.6.7 for detailed requirements and examples.

— **Rule 3:** Only the events requested shall be returned in a response. For example, if only class 1 and class 2 events are requested, the outstation shall not include class 3 events in the response.

NOTE — Be aware that it is possible to report events out of order if an outstation assigns some binary input events to class 1 and others to class 2, as an example, and the master polls for only class 1 or only class 2 events. This is because events in the non-polled class may have been generated at an earlier time than those reported in the polled class.

— **Rule 4:** An event should be reported only once within a response. If an event happens to meet the criteria appearing in two or more request objects, only report the event once—do not duplicate it.

NOTE — Multiple events can originate from the same point. For example, a binary input changes from 0 to 1 and then within milliseconds changes from 1 back to 0. In this example, two **different** events would have been generated and both may be included in the same response. Rule 4 applies to a single event.

— **Rule 5:** If event data is requested, the outstation shall place it in the response ahead of any static data values that were also requested in the same request. Requests to read non-event data should be processed in the order of the object headers and object prefixes appearing in the request.

— **Rule 6:** A master shall process each DNP3 object returned in a response in the order that it appears in the received fragment. This causes the sequence of changes appearing in the master's database to correspond to the sequence order observed in the field, and the final state of each database point should contain the most recently occurring event state. See Clause 5 for additional details.

4.4.2.2 Examples

A few specific examples follow. These show a tiny fraction of all possible ***read*** requests and responses; their purpose is to illustrate various features of the protocol. Only enough details of the actual DNP3 objects are given in the examples to help the reader understand the concepts. Complete details are provided in Annex A. Message **octet contents are shown in hexadecimal** in the rows immediately below the gray lines.

EX 10	This example shows a request for the static analog values from indexes 4 through 7 returned as a 16-bit value with a flag octet.

►►► Request Message

C3	01	1E	02	00	04	07
AC	FC	Grp	Var	Qual	Range	

◄◄◄ Response Message (beginning)

C3	81	00	00	1E	02	00	04	07	01	88
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range		Flg_4	LSB_4
									←	DIO_4

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

◄◄◄ Continuation of Response Message

13	01	20	4E	01	50	FB	01	60	00
MSB_4	Flg_5	LSB_5	MSB_5	Flg_6	LSB_6	MSB_6	Flg_7	LSB_7	MSB_7
→	←	DIO_5	→	←	DIO_6	→	←	DIO_7	→

In the example, all the flag octets indicated an on-line status and no exceptions or errors. Therefore, because the absence of a flag is equivalent to on-line and no exceptions, the following response is just as valid.

◄◄◄ Response Message (beginning)

C3	81	00	00	1E	04	00	04	07	88	13
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range		LSB_4	MSB_4
									← DIO_4 →	

◄◄◄ Continuation of Response Message

20	4E	50	FB	60	00
LSB_5	MSB_5	LSB_6	MSB_6	LSB_7	MSB_7
← DIO_5 →		← DIO_6 →		← DIO_7 →	

Because all the flags would indicate an on-line status and no exceptions or errors, it is permissible to omit the flag octets. The variation number was changed from 02 to 04. The message without the flag octets is four octets shorter—a 20% savings in this example!

EX 11	This example shows a request for all of the static binary inputs. Assume there are 18 binary inputs.

►►► Request Message

C3	01	01	01	06
AC	FC	Grp	Var	Qual

Note that no range field appears when the qualifier 06 is specified; it means values from all points.

◄◄◄ Response Message (beginning)

C3	81	00	00	01	01	01	00	11	0F	AA
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range		←	States

◄◄◄ Continuation of Response Message

03
→

The states of the first eight binary inputs appear in the first value octet. In this example, the first octet contains 0F, which means that the state is 1 for indexes 0 through 3, and that the state is 0 for indexes 4 through 7. The states of indexes 8 through 15 alternate between 0 and 1. The last two states, indexes 16 and 17, are 1's. The remainder of the last octet is padded with six 0 bits to complete the octet.

NOTE

Qualifier 06, meaning all objects, cannot be used in a response because there would not be enough information to explicitly identify the indexes to the receiver.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

EX 12	This example illustrates a request to read class 1 and class 2 event data. Event data is requested with object group 60.

►►► Request Message

C3	01	3C	02	06	3C	03	06
AC	FC	Grp	Var	Qual	Grp	Var	Qual

The reader should note that this is a multiple-object request because more than one object header appears in the request, namely a class 1 and a class 2 object header.

NOTE

> Only qualifiers 06, 07, and 08 may be used to request events because the requester does not know which indexes have events until the response is returned. Qualifier 06 specifies to send all of the events, and qualifiers 07 and 08 specify the maximum number of events to place in the response—fewer events may be returned if the outstation has less than the quantity requested.
>
> While qualifier 09 is valid, it should be avoided—see 4.2.2.7.3.4.

The response, in this example, returns 3 binary input changes and 1 analog value change. Binary input events from indexes 3 and 20 are from class 1, binary input from index 5 is from class 2, and analog input 11 is from class 2. (The index assignments are arbitrarily for use with this example only.)

◄◄◄ Response Message (beginning)

E3	81	06	00	02	01	17	01	14	81	02
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Index	Value	Grp
				←	1st Event				→	←

◄◄◄ Continuation of Response Message

01	17	01	05	01	20	02	17	01	0B	20
Var	Qual	Range	Index	Value	Grp	Var	Qual	Range	Index	Flg
	2nd Event			→	←			3rd Event		

◄◄◄ Continuation of Response Message

FF	FF	02	01	17	01	03	81
Value		Grp	Var	Qual	Range	Index	Value
	→	←		4th Event			→

There are four things worth noting in this response. The first is that the response does not identify the event class.

The second is that bit 5 is set in the application control octet; this directs the master to send an Application Layer confirmation message back upon receipt of this response.

NOTE

> Outstations should request Application Layer confirmation in all responses that contain event data. When a confirmation with the same sequence number is received from the master, then event data for the events that were transmitted can be removed from the outstation's buffers.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

The next issue of note is that the events are not sent in class or index order. Clause 5 specifies event reporting order requirements. In general, events are returned in the order that they occurred to permit the master to reconstruct the history and proper ordering of states and values. The reader should notice that in this example, the event variations did not include time stamps.

The fourth observation is that the response could have used a single object header for the first and second event. The same group, variation, and qualifier were used, the range could have specified two objects, and the second header could have been omitted. Outstation devices with enough computing power are encouraged to optimize the responses, when possible, to conserve bandwidth.

►►► Confirm Message

C3	00
AC	FC

Rule 3, 4.4.2.1, puts the onus on the master to formulate requests properly so that its application software receives the required information. If the outstation in the previous example had had a class 3 event that occurred prior to the other events in the response, it would not have been transmitted because only class 1 and class 2 events were requested. Therefore, the master could not know that the class 3 event occurred first.

EX 13	This example shows the master requesting a maximum of 20 binary events. When a master requests event data using a group number other than 60 (class data), as this example does, it expects to receive events from the respective point type without regard to the event class assignments for any of those points.

►►► Request Message

C3	01	02	00	07	14
AC	FC	Grp	Var	Qual	Range

The request uses variation 0. Variation 0 has special meaning, and only master requests may use it. Variation 0 indicates that the master does not have a preferred format for the outstation to use in its response—the outstation determines which object variations to return.

◄◄◄ Response Message (beginning)

E3	81	06	00	02	01	17	03	14	81	05
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Index	Value	Index
								1st Event		2nd

◄◄◄ Continuation of Response Message

01	03	81
Value	Index	Value
Event	3rd Event	

This response is somewhat similar to the previous example's response, but this only contains binary input events and no analog input events. It also uses a common object header for efficiency. The same requirements for time sequence order apply. If there had been a binary input event in class 3, it too would have appeared.

It is worth noting that even though the master requests a specific count of events, the count represents a maximum number, and it is acceptable to return less if the outstation does not have the requested number of events. Refer to 4.2.2.7.3.3 for more details when range qualifier codes 0x07, 0x08, and 0x09 are used.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

►►► Confirm Message

C3	00
AC	FC

EX 14	This example illustrates returning what is called a "null response" because no data objects are included. Outstations send null responses when the master requests events and no events qualify for reporting back in a response.

►►► Request Message

C3	01	02	00	07	14
AC	FC	Grp	Var	Qual	Range

This request is identical to the previous example's request.

◄◄◄ Response Message (beginning)

C3	81	04	00
AC	FC	IIN_1	IIN_2

This is the null response. There were no binary events, and none were reported so only the application control octet, function code octet, and internal indications octets were reported. It is still possible for analog input events to exist, but because they did not qualify according to the request (namely binary input events), they cannot appear in the response.

NOTE A **null response** consists of only the application control, function code, and internal indications octets.

Also observe that an Application Layer confirmation was not requested in the response (i.e., bit 5 was not set). The outstation does not require confirmation because it does not discard any events from its event buffer(s).

4.4.3 Function code 2 (0x02)

Write

The WRITE function code is complementary to the READ function code. Writing copies the contents of DNP3 objects from the master to the outstation. Some uses for writing are clearing bit IIN1.7 [DEVICE_RESTART], setting time in the outstation, setting analog deadband values, and downloading configuration files.

The object headers in the request specify which objects the master desires to write. The respective data information objects follow each object header.

In most, but not all cases, ***write*** responses are null responses (i.e., they only contain the application control octet, function code octet, and internal indications octets). Writing file data is one of the exceptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

The general formats of ***write*** request and response messages are shown as follows.

AC is the Application Control octet.

FC is the Function Code octet.

$IIN_{1,2}$ represents the Internal Indication octets.

OH_x is the xth Object Header. (These are shown shaded.)

DIO_x is the xth Data Information Object.

►►► Request Message (beginning)

AC	FC	OH_0	DIO_0	DIO_1	• • •	DIO_i	OH_1	DIO_0	DIO_1	• • •

►►► Continuation of Request Message

DIO_j	• • •	OH_n	DIO_0	DIO_1	• • •	DIO_k

◄◄◄ Response Message

AC	FC	$IIN_{1,2}$

4.4.3.1 Rules

— **Rule 1:** Masters shall not retry **write** request messages at either the Application or Data Link Layers for time synchronization where the messages contain either an Absolute Time object, g50v1, or a Last Recorded Time object, g50v3. For more information, see Clause 10.

4.4.3.2 Examples

This clause provides a few ***write*** examples.

EX 15	This example shows a master clearing an outstation's IIN1.7 [DEVICE_RESTART] bit.

►►► Request Message

C3	02	50	01	00	07	07	00
AC	FC	Grp	Var	Qual	Start	Stop	Value
					Range		

◄◄◄ Response Message

C3	81	00	00
AC	FC	IIN_1	IIN_2

EX 16	This example shows setting analog deadbands for indexes 6, 8, and 20.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

►►► Request Message

C3	02	22	01	17	03	06	12	00	08	4A
AC	FC	Grp	Var	Qual	Range	Index	Value		Index	Value
						←	DIO_6	→	←	DIO_8

►►► Continuation of Request Message

00	14	FF	FF
Value	Index	Value	
→	←	DIO_{20}	→

◄◄◄ Response Message

C3	81	00	00
AC	FC	IIN_1	IIN_2

EX 17	This example shows setting the time.

►►► Request Message

C3	02	32	01	07	01	AC	E9	00	40	08
AC	FC	Grp	Var	Qual	Range	Time				

►►► Continuation of Request Message

01
Time

◄◄◄ Response Message

C3	81	00	00
AC	FC	IIN_1	IIN_2

4.4.4 Function codes 3 (0x03) *and* 4 (0x04)

Select *and* Operate

The SELECT function code is used in conjunction with the OPERATE function code as part of the two-step, select-before-operate method for issuing control requests. This procedure is used for controlling binary[10] and analog outputs.

Requests with these function codes contain one or more objects that describe the desired output state or level.

An outstation's ***select*** and ***operate*** responses contain the identical set of object headers and objects, and in the same order as they appear in the master's request, unless the outstation determines an error condition

[10] Some output objects in the Object Library are not suitable for use with select–operate function codes because they do not provide a means for reporting errors. An example is a g10 v1.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

exists. In the case of an error, an outstation sets a status code within an object, possibly omits objects from the response, and/or sets IIN bits as described elsewhere in this specification.

4.4.4.1 Select–operate philosophy

The general select–operate procedure is for the master to first send a ***select*** request containing all of the necessary parameters, such as indexes, timings, and values. The ***select*** response from the outstation contains object headers and objects that exactly match those in the request. The master compares the ***select*** response with the request, and if they match exactly, then the master issues an ***operate*** command with the identical object set of headers and objects that it sent in the ***select*** message. This approach assures, with miniscule probability of error, that the outstation understands which control the master intended. There are two verifications during a complete, two-step procedure:

— The master receives a **select** response that matches the request, and that response indicates no errors; otherwise, the master shall abort the control.

— The outstation is obligated to compare the **operate** request with the **select** request, and only if the two match, and there are no errors detected in the request, should it activate the outputs.

4.4.4.2 Multiple control objects

Select and ***operate*** requests may contain multiple objects if the outstation supports having multiple control objects in the same request. For example, it is permissible to send two or more CROBs, g12v1, in ***select*** and ***operate*** messages, or two or more analog output blocks (AOBs), group 41, any variation, may appear in ***select*** and ***operate*** messages.

4.4.4.2.1 CROBs and AOBs

Whenever the master sends select–operate commands with multiple objects, it is intended that the outstation shall execute them expeditiously. The master should not assume the points are executed simultaneously or in sequential order as the implementation details are private to the outstation. Although some outstations may be capable of executing the controls simultaneously, others cannot and shall queue the objects for execution one-after-the-other.

Outstations are not required to support more than one DNP3 control object per request message.

NOTE | The number of CROBs and AOBs that an outstation can accept in one message should be listed in its device profile document. A statement should also appear in the outstation's device profile document indicating whether CROBs and AOBs are permitted in the same request.

4.4.4.2.2 Pattern Control Blocks and Masks

If the master desires simultaneous execution of controls, it should use Pattern Control Block and Pattern Mask objects, g12v2 and g12v3, respectively. For multiple controls to execute simultaneously, a Pattern Control Block and Pattern Mask shall be used, **and** the outstation supports this type of operation.

Outstations are not required to implement Pattern Control Blocks, g12v2, or Pattern Masks, g12v3. The outstation's device profile document shall indicate whether it supports these.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.4.4.3 Control-related rules

— **Rule 1:** Outstations shall start a selection timer upon receipt of a valid **select** request. If that timer times out prior to receiving a valid **operate** request, the outstation shall reject the operate request and return a status code indicating the timer is expired.

— **Rule 2:** An outstation shall compare all of the octets following the function code octet in the **operate** request with all of the octets following the function code octet in the **select** request. The outstation shall only activate, or queue for operation, the specified point or points if the octets match exactly, and in its response to the corresponding **select** request, the outstation would **not**

1) Return a non-zero status code in any of the objects.
2) Set IIN bit 2.0 [NO_FUNC_CODE_SUPPORT].
3) Set IIN bit 2.1 [OBJECT_UNKNOWN].
4) Set IIN bit 2.2 [PARAMETER_ERROR].
5) Set IIN bit 2.4 [ALREADY_EXECUTING].

— **Rule 3:** If an outstation returns a non-zero status code in any object in its response to a **select** request, the outstation shall cancel the entire selection.

— **Rule 4:** If a selection is in effect at the outstation, upon receipt of the next request, the outstation shall perform the action as listed in Table 9 depending on the function code and sequence number in the application control octet.

— Keys for understanding table:

N is the sequence number from the *SEQ* field in application control octet of the original selection message.

!= means “Not Equal To”; e.g., !=N means the sequence number is not equal to N.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 9—Action to perform with next request after a select request

Next request contains			Outstation action
Request function code	**Sequence number modulo 16**	**Octets in the message after the function code octet match those in the original select request**	
select	N	Yes	This is a valid retry. Repeat the response to the original ***select*** request—do not restart the selection timer.
select	N	No	Discard fragment. Take no further action.
select	!=N	No	This is a new selection that overrides the current selection. Terminate the original selection, initiate a new selection, and restart selection timer.
operate	N + 1	Yes	Check for other errors or other discrepancies, and if none are found, initiate the control execution.
operate	N + 1	No	Non-matching octets. Terminate original selection and perform no control action.
operate	!=(N + 1)	Don't Care	Improper sequence number. Terminate original selection and perform no control action.
Neither select nor operate	Don't Care	Don't Care	Operate did not follow select. Terminate original selection and perform no control action.

—

— **Rule 5:** When multiple control objects are included in **select** and **operate** messages, the outstation has the option to stop parsing the remainder of a request upon detection of the first error or continue to the end of the request.

— **Rule 6:** When an outstation receives a **select** or **operate** request, and operations to one or more of the points specified by the objects in the request are unsuccessful, its response shall include one of these

1) No DNP3 objects if the request contains an unsupported function code or object variation.
2) All of the DNP3 objects.
3) Only objects for all of the points up to and including the first unsuccessful point. This is called a "truncated response."

— **Rule 7:** Each DNP3 object returned in the response shall contain the appropriate status code indication. The IIN bits 2.0, 2.1, 2.2, and 2.4 shall be set or cleared as applicable. The IIN bits only need to reflect the state of the first error detected.

— **Rule 8:** A master shall compare all of the octets following the two IIN octets in the **select** response with all of the octets following the function code octet in the request. If the octets do not match exactly or any of these conditions are true

1) the status code returned in any object is non-zero,
2) IIN bit 2.0 [NO_FUNC_CODE_SUPPORT] is set,
3) IIN bit 2.1 [OBJECT_UNKNOWN] is set,
4) IIN bit 2.2 [PARAMETER_ERROR] is set,

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5) IIN bit 2.4 [ALREADY_EXECUTING] is set,

then the master shall not issue the matching **operate** command.

NOTE

> The DNP3 requirements were less strict in versions of this document prior to 2.00. For the sake of backward compatibility, the outstation has two options if a master illegally issues an ***operate*** command after receiving an outstation's ***select*** response in which the response had missing objects (truncated response; see Rule 6), non-zero status codes or IIN bits 2-0, 2-1, 2-2, or 2-4 set. The outstation may
>
> — Choose to ignore the request and not operate any of the outputs.
>
> — Execute those points for which it returned objects with a zero status code.
>
> The outstation's behavior is not guaranteed when the master violates Rule 8.

— **Rule 9:** The master shall not retry the select–operate sequence in its entirety. If the select portion succeeds but the operate portion fails, a master shall not retry the **select- operate** requests as this can result in duplicate controls. The definition of a select–operate retry is sending **select** and **operate** requests using the same sequence numbers in the application control octet as were in the **select** and **operate** requests of the failed attempt. If a repeated operation is acceptable or desired, the master should send similar, but new messages with the sequence numbers properly incremented.

4.4.4.4 Examples

This subclause contains various control examples.

EX 18	This example shows ***select–operate*** messages for a CROB (g12v1) issued to point 10. The command is to close the point one time for 250 milliseconds.

In the first exchange, the master sends a request with the SELECT function code.

►►► **Select** Request Message (beginning)

C3	03	0C	01	17	01	0A	41	01	FA	00
AC	FC	Grp	Var	Qual	Range	Prefix	←		CROB	

►►► Continuation of **Select** Request Message

00	00	00	00	00	00	00	
		CROB continued				→	

◄◄◄ **Select** Response Message (beginning)

C3	81	00	00	0C	01	17	01	0A	41	01
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Prefix	←	

◄◄◄ Continuation of **Select** Response Message

FA	00	00	00	00	00	00	00	00	
			CROB					→	

The octets in ***select*** response message after the IIN octets match the octets in the ***select*** request after the function code. This is the expected response; therefore, the master may send an ***operate*** request message.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

►►► **Operate** Request Message (beginning)

	C4	04	0C	01	17	01	0A	41	01	FA	00
|	AC |	FC |	Grp |	Var |	Qual |	Range |	Prefix |	←		CROB	

►►► Continuation of **Operate** Request Message

00	00	00	00	00	00	00	
		CROB continued				→	|

◄◄◄ **Operate** Response Message (beginning)

	C4	81	00	00	0C	01	17	01	0A	41	01
|	AC |	FC |	IIN_1 |	IIN_2 |	Grp |	Var |	Qual |	Range |	Prefix |	←	

◄◄◄ Continuation of **Operate** Response Message

FA	00	00	00	00	00	00	00	00	
			CROB					→	|

The ***operate*** request message is almost identical to the ***select*** message, except the sequence number in the application control octet is incremented by one, and the function code is OPERATE. The outstation shall compare the ***operate*** and ***select*** messages, and if all of the octets following the function code octets match, it is permitted to actuate the output.

EX 19	This example illustrates the use of a Pattern Control Block (PCB) and a Pattern Control Mask (PCM). Points 5, 10, and 15 are activated three times, each for 500 milliseconds with a 2 second period between actuations.

In the first exchange, the master sends a request with the SELECT function code. Notice that the PCB appears first in the message, and the PCM follows it.

►►► **Select** Request Message (beginning)

	C3	03	0C	02	07	01	41	03	F4	01	00
|	AC |	FC |	Grp |	Var |	Qual |	Range |	←		PCB		

►►► Continuation of **Select** Request Message

00	D0	07	00	00	00	0C	03	00	05	0F
		PCB continued			→	| Grp |	Var |	Qual |	Range	|

►►► Continuation of **Select** Request Message

21	04	
| PCM		|

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

◄◄◄ **Select** Response Message (beginning)

C3	81	00	04	0C	02	07	01	41	03	F4
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	←	PCB	

◄◄◄ Continuation of **Select** Response Message

01	00	00	D0	07	00	00	04	0C	03	00
PCB continued							→	Grp	Var	Qual

◄◄◄ Continuation of **Select** Response Message

05	0F
Range	

This time, the control command is not issued because an error was reported in the status code of the PCB. It indicated that the index for one of the points was beyond the number configured. IIN2.2 [PARAMETER_ERROR] was also set specifying a parameter error. Notice that the PCB status code indicated the error rather than the PCM because a pattern mask does not have its own status code.

EX 20	Table 10 shows example responses from CROB or analog output requests containing four objects. These objects are identified as A, B, C and D. The responses are identical for requests having the SELECT, OPERATE or DIRECT_OPERATE function codes.

Keys: NR means Not Reported.

Status codes are defined in Annex A.

Some of the responses show two sets of choices for the object status codes; either is acceptable.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 10—Example status codes and IIN bits in control response

Conditions in outstation when request is received	Response								
	Status codes				IIN bits				
	Obj A	Obj B	Obj C	Obj D	1.5	2.0	2.1	2.2	2.4
No errors are detected; all points are successful.	0	0	0	0	0	0	0	0	0
The function code is not supported regardless of which indexes the objects have.	NR	NR	NR	NR	0	1	0	0	0
The outstation does not support the specific variation code in the request.	NR	NR	NR	NR	0	0	1	0	0
Indexes for objects C and D are beyond the maximum number of points installed in the outstation.	choice 1 0	0	4	4	0	0	0	1	0
	choice 2 0	0	4	NR					
The point in object B is already executing when this request arrives.	choice 1 0	5	0	0	0	0	0	0	1
	choice 2 0	5	NR	NR					
The outstation is not able to control more than one point at a time.	choice 1 0	8	8	8	0	0	0	0	0
	choice 2 0	8	NR	NR					
The point in object C is tagged or blocked to prevent its control.	choice 1 0	0	9	0	0	0	0	0	0
	choice 2 0	0	9	NR					
The Remote/Local Switch is in the Local position.	choice 1 7	0	7	7	1	0	0	0	0
	choice 2 7	NR	NR	NR					
A control output is requested, and the CROB in object D's request contains an illegal control code.	0	0	0	3	0	0	0	0	0
An analog output is requested, and the value in object D's request exceeds the permitted level.	0	0	0	3	0	0	0	0	0

4.4.5 Function codes 5 (0x05) *and* 6 (0x06)

Direct Operate *and* Direct Operate—No Acknowledgment

These direct operate functions are similar to the OPERATE function code except that no preceding ***select*** command is required. They are used for outputting CROBs, Pattern Control Blocks, and analog outputs when the extra security provided by a two-step control command is not necessary. Another use is to optimize bandwidth utilization in closed-loop control when other feedback is present.

Direct operate request messages look identical to ***select*** and ***operate*** request messages except for the function code. They contain one or more objects that describe the desired output state or level.

Direct operate responses contain the identical set of object headers and objects, and in the same order as appear in the master's request unless it determines an error condition exists. In the case of an error, the

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

outstation sets a status code within an object, possibly omits objects, and/or sets IIN bits as described elsewhere in this specification.

The discussion in 4.4.4.2 regarding multiple objects applies to this function code. The operate messages in the examples in 4.4.4.4 properly illustrate direct operate messages.

The response to a DIRECT_OPERATE command does not verify that execution actually occurred. It only indicates that the request was received. For this reason, systems employing either of these function codes are encouraged to provide another means for detecting that execution did happen.

The DIRECT_OPERATE_NR function code is similar to the DIRECT_OPERATE function code except that the outstation does not send a response message; it does, however, execute the control if no errors are detected in the request. This function code is suitable for broadcasting a control request from one master to multiple outstations, where each outstation is identically equipped. It is important to realize that when employing function code DIRECT_OPERATE_NR in a request, there is no direct feedback for assuring that an error-free request was received.

4.4.5.1 Rules

There are a few rules regarding direct operate functionality

— **Rule 1:** Every master and outstation device that supports a direct operate operation shall also provide support for the select–operate operation. This allows the system owner or user to choose the security level appropriate for the installation.

— **Rule 2:** A master shall never retry sending a message with a DIRECT_OPERATE function code as this can result in duplicate control actions when, unknown to the master, an outstation restarts. The definition of a retry is a repeated request having the same sequence number in the application control octet as the previous request. If a repeated operation is acceptable or desired, the master should send a similar, but new, message with the sequence number properly incremented.

4.4.6 Function codes 7 (0x07) *and* 8 (0x08)

Immediate Freeze *and* Immediate Freeze—No Acknowledgment

The purpose of this function is to copy the current value of a counter or analog point to a second, separate memory location associated with the same point. The copied value is referred to as the frozen value and remains constant until the next freeze operation to the same point. These commands do not affect the current values of the counter or analog points.

NOTE

It is often important to simultaneously freeze all of the counts throughout an entire system at regular times to obtain snapshots in time of the counts at every device in the system.

For the IMMED_FREEZE function code, the response is a null response. For the IMMED_FREEZE_NR function code, no response is sent, and for this reason, the IMMED_FREEZE_NR function code is recommended for broadcast freezing.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.4.6.1 Objects in freeze requests

Annex A contains two point types that are freezable, counters and analog inputs. Although an outstation device is not required to have freezable points, some DNP3 implementation levels require that the outstation be able to parse freeze messages.

Freeze requests are sent using object headers for the current value object group (20 or 30). Frozen, snapshot values are read using object headers for the frozen object group (21 or 31). Note that the object groups in freeze commands are different than the object groups used to read the frozen values.

The object variation for all points specified in a freeze request is **always zero** because freezing does not apply to any particular data format and no DNP3 objects are returned in the response.

4.4.6.2 Examples

EX 21	In this example, a master requests the outstation to freeze all of its counters.

►►► Request Message

C3	07	14	00	06
AC	FC	Grp	Var	Qual

◄◄◄ Response Message

C3	81	00	00
AC	FC	IIN_1	IIN_2

4.4.7 Function codes 9 (0x09) *and* 10 (0x0A)

Freeze-and-Clear *and* Freeze-and-Clear—No Acknowledgment

These function codes are similar to function codes IMMED_FREEZE and IMMED_FREEZE_NR in all respects except that after copying the current value to the frozen value, the current value is immediately cleared to 0.

An example is a counter that accumulates pulses and is frozen at periodic time intervals. At each freeze, the counter is cleared and resumes its counting from zero. In an electrical system, if the counts represent energy, then the frozen value represents demand. In a water system, if the counts represent volume, then the frozen count represents a flow rate.

For function code FREEZE_CLEAR, the response is a null response. For function code FREEZE_CLEAR_NR, no response is sent, and for this reason, function code FREEZE_CLEAR_NR is recommended for broadcast freezing.

See 4.4.6.1 regarding which objects appear in requests with these function codes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.4.8 Function codes 11 (0x0B) *and* 12 (0x0C)

Freeze-at-Time *and* Freeze-at-Time—No Acknowledgment

These function codes initiate periodic freezing of the specified points. The request message contains a Time-Date-and-Interval object header and object, g50v2, followed by object header(s) for the point(s) that are to obey the freezing schedule. Multiple schedules may be sent in the same request. Upon receipt of this type request, an outstation automatically performs the freeze operations according to the schedule without further commands from the master station. The schedule may require the outstation to perform either a single freeze operation or an infinite number of freeze operations.

The response is a null response.

See 4.4.6.1 regarding which objects appear in requests with these function codes.

The general formats of function code FREEZE_AT_TIME or FREEZE_AT_TIME_NR requests and the FREEZE_AT_TIME function code response messages are shown as follows. No response is sent with function code FREEZE_AT_TIME_NR.

AC is the Application Control octet.

FC is the function code octet.

IIN_1 and IIN_2 represent the Internal Indication octets.

TDH_x is the xth **T**ime-**D**ate-and-Interval object **H**eader (shown shaded).

TDOx is the xth **T**ime-**D**ate-and-Interval DNP3 **O**bject (shown shaded).

DOH_{xy} is the yth **D**ata **O**bject **H**eader specifying a point that is to be frozen according to schedule in the xth Time-Date-and-Interval object.

►►► Request Message (beginning)

AC	FC	TDH_0	TDO_0	DOH_{00}	DOH_{01}	• • •	DOH_{0i}	TDH_1	TDO_1	DHO_{10}

►►► Continuation of Request Message

DHO_{11}	• • •	DHO_{1j}

◄◄◄ Response Message

AC	FC	IIN_1	IIN_2

A Time-and-Date-with-Interval DNP3 object has a time-date field and an interval field. These two fields are used in a binary code-like scheme to indicate when to freeze the points. This is shown in Table 11.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 11—Freezing schedule interpretation

Time-date field	Interval field	Freeze timing
zero	zero	Freeze once immediately.
non-zero	zero	Freeze once at the specified time.
zero	non-zero	Periodically freeze at intervals relative to the beginning of the current hour. Use the time interval from the interval field. Continue freezing forever or until a new function code FREEZE_AT_TIME or FREEZE_AT_TIME_NR freeze request is received.
non-zero	non-zero	Periodically freeze at intervals relative to the time and date in the time-date field. Use the time interval from the interval field. Continue freezing forever or until a new function code FREEZE_AT_TIME or FREEZE_AT_TIME_NR freeze request is received.

NOTE To cancel a freeze command, set the time-date field to all ones.

4.4.8.1 Examples

EX 22	This is an example of freezing all counters beginning on 15 June 2001 at 14 hours, 2 minutes, 0 seconds and 0 milliseconds, and every 15 minutes thereafter.

►►► Request Message (beginning)

C3	0B	32	02	07	01	C0	5F	63	1C	E7
AC	FC	Grp	Var	Qual	Range	←	Time & Date w/ Interval Object			

►►► Continuation of Request Message

00	A0	BB	0D	00	14	00	06
T & D w/ Intvl Obj Continued				→	Grp	Var	Qual

◄◄◄ Response Message

C3	81	00	00
AC	FC	IIN_1	IIN_2

4.4.9 Function codes 13 (0x0D) *and* 14 (0x0E)

Cold Restart *and* Warm Restart

A COLD_RESTART function code forces the outstation to perform a complete restart similar to what the device would do upon powering up after a long-term power loss. Many devices clear all output hardware to a safe state and re-initialize themselves with configuration information and/or default values and clear all queues. The specific actions performed are device dependent and not defined herein.

A WARM_RESTART function code forces the outstation to perform a partial reset. Only the DNP3 application needs to reset, and no affect to other subsystems and processes within the outstation is required.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Some devices re-initialize the DNP3 application with configuration information and/or default values and clear all event and control queues. When practical, control operations in progress are terminated. The specific actions performed are device dependent and not defined herein.

When an outstation receives a ***cold restart*** or ***warm restart*** request, it shall immediately respond with a Delay Time DNP3 object, g52v1 or g52v2, and then initiate its restart activities. The delay time in the object specifies the length of time during which the outstation expects to be busy and unable to respond to requests.

NOTE | The master should honor the time delay in the response and not attempt to send any requests until the period has elapsed.

4.4.9.1 Examples

EX 23	This is an example of a cold restart where the response indicates 45 seconds.

►►► Request Message

C3	0D
AC	FC

◄◄◄ Response Message

C3	81	00	00	34	01	07	01	2D	00
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Time Delay	

4.4.10 Function code 15 (0x0F)

Initialize Data (Obsolete)

This function code is obsolete, and new designs shall not implement it.[11]

Originally, it was intended to tell the outstation to set configurable data to the default, or initial startup, settings. The response to an ***initialize data*** request is a null response.

4.4.11 Function codes 16 (0x10) *and* 17 (0x11) *and* 18 (0x12)

Initialize Application *and* Start Application *and* Stop Application

These function codes initialize, start, and stop the application(s) specified in the request. Applications are unique to the outstation device and not defined in the DNP3 protocol. A local closed-loop control is an example of such an application.

[11] The reason for deprecating function code 15 was because versions of this specification prior to 2.00 did not provide sufficient details about how to use the function code or clearly define the behavior of an outstation upon receipt of such a request.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Applications are specified in the request with Application Identifier objects using object group 90. When referencing a specific application in a message, the object qualifier octet shall be 0x5B. Qualifier 0x06 is used to specify all applications without identifying a specific one.

The response messages for each of the three function codes are null messages.

The ***initialize application*** function is optional for an outstation application and depends on the requirements of the specific application in the outstation. Nevertheless, even though there may be nothing to initialize, outstations that have DNP3-controllable applications shall parse and respond to this function.

NOTE An alternative to using these function codes is for the outstation to implement pseudo binary control points that correspond to the application.

4.4.11.1 Examples

EX 24	This is an example of how to control an outstation application named CL6. Application identifier objects do not have a defined format. For this example, the application name, "CL6," is used.

►►► Request Message – Initialize Application

C3	10	5A	01	5B	1	03	00	43	4C	36
AC	FC	Grp	Var	Qual	Range	Size Prefix		'C'	'L'	'6'

◄◄◄ Response Message

C3	81	00	00
AC	FC	IIN_1	IIN_2

►►► Request Message – Start Application

C3	11	5A	01	5B	1	03	00	43	4C	36
AC	FC	Grp	Var	Qual	Range	Size Prefix		'C'	'L'	'6'

◄◄◄ Response Message

C3	81	00	00
AC	FC	IIN_1	IIN_2

• • • Application running • • •

►►► Request Message – Stop Application

C3	12	5A	01	5B	1	03	00	43	4C	36
AC	FC	Grp	Var	Qual	Range	Size Prefix		'C'	'L'	'6'

◄◄◄ Response Message

C3	81	00	00
AC	FC	IIN_1	IIN_2

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.4.12 Function code 19 (0x13)

Save Configuration

This function specifies that the outstation should store into non-volatile memory the contents of a configuration file located in volatile memory.

When an outstation receives a Save Configuration request, it shall immediately respond with a Delay Time DNP3 object, g52v1 or g52v2, and then initiate its storage activities.

Function code SAVE_CONFIG is deprecated, and new designs shall avoid using it.[12] Function code ACTIVATE_CONFIG was added, and in some situations, it may be a suitable replacement.

4.4.13 Function code 20 (0x14) *and* 21 (0x15)

Enable Unsolicited Responses *and* Disable Unsolicited Responses

A master uses these functions to dynamically enable and disable which points may, or may not, be included in a spontaneous, unsolicited message.

Outstations that do not support unsolicited messages are not obligated to implement these functions and may ignore the remainder of this clause. Outstations that do support unsolicited messages shall implement these two function codes.

An outstation may only include event objects in an unsolicited response message from points that have been enabled by an ***enable unsolicited responses*** request. Events that were generated before a point is enabled shall not be reported in unsolicited responses until after the point is enabled and if those events have not already been read and confirmed by a master poll (solicited request, response and confirmation).

As a minimum, an outstation shall accept commands to enable and disable unsolicited responses by event class (using object headers with group number 60) even if the device does not have class 1, 2, or 3 data when the request arrives.

Enabling and disabling unsolicited messages on a per point type and index is optional. When this is implemented, the object headers in the request message specify the group number corresponding to **static** data of a point type and variation 0—the request message shall not use **event** type object headers. Masters shall send variation 0, but to accommodate legacy systems, outstations may ignore the variation number. An object header and index list as shown in Figure 12 and its accompanying description may be used in these requests.

Regardless of the cause, when an outstation is reset or restarted, all of its points shall be disabled from initiating unsolicited responses. This does not mean the points do not generate events, just that the points cannot initiate unsolicited reporting. An outstation shall not report unsolicited events until its points are explicitly enabled by a request from the master, and then only data from the enabled points are permitted to be included in the response. Note that devices shall transmit a data-less null message upon restarting according to the requirements in Clause 5 even though no points are enabled at the time.

When an outstation receives a function code DISABLE_UNSOLICITED request to disable initiation of unsolicited responses from points identified by the object headers in the request, it shall no longer transmit

[12] The reason for deprecating function code 19 was because versions of this specification prior to 2.00 did not provide sufficient details about how to use the function code or clearly define the behavior of an outstation upon receipt of such a request.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

any data via an unsolicited response for those points. The request also cancels any pending expectation of confirmation for an unsolicited response that has already been sent from the outstation, but for which confirmation has not yet been received.

An outstation shall not lose or discard event data as a result of receiving the DISABLE_UNSOLICITED function code; the outstation shall report events if they are requested in a master poll for those points that were disabled from reporting in unsolicited responses.

The response to ***enable unsolicited*** and ***disable unsolicited*** requests are null responses.

4.4.13.1 Examples

EX 25	This is an example of enabling an outstation to transmit all class 1, 2, and 3 events.

►►► Request Message

C3	14	3C	02	06	3C	03	06	3C	04	06
AC	FC	Grp	Var	Qual	Grp	Var	Qual	Grp	Var	Qual

◄◄◄ Response Message

C3	81	00	00
AC	FC	IIN_1	IIN_2

4.4.14 Function code 22 (0x16)

Assign Class

A master uses this function code to assign the events generated by points to event classes. Present assignments may be altered using this function code.

Every device shall have a default event class for each and every point for which it supports events. This is necessary so that when a master requests events by class, an outstation only reports events from those points whose class assignments match the request. It is desirable, but not mandatory, for devices to provide a means of configuring the default class assignments for each of its points.

When an event is created, it is classified as a class 1, 2, or 3 event. The convention used by DNP3 to disable event generation is to specify an assignment to class 0. The events are not really assigned to class 0 because that class specifies static data, but using class 0 in the request allows DNP3 to employ a consistent object format.

Static data is always assigned to class 0 and is not re-assignable to any other class.

The request contains one or more **sets** of assignments. Each assignment **set** contains one class object header followed by data object headers that specify the points whose events are to be re-assigned.

AC is the Application Control octet.

FC is the function code octet.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

IIN_1 and IIN_2 represent the Internal Indication octets.

COH_x is the xth **C**lass **O**bject **H**eader (shown shaded).

DOH_{xy} is the yth **D**ata **O**bject **H**eader associated with the xth class object header.

►►► Request Message

AC	FC	COH_0	DOH_{00}	DOH_{01}	• • •	DOH_{0n}	COH_1	DOH_{10}	• • •	DOH_{1m}

◄◄◄ Response Message

AC	FC	IIN_1	IIN_2

Within each assignment **set**

— The variation octet in the class object header specifies the new event class.

— All of the object headers that follow the class object header, until the next class object header or the end of the request, specify the points that shall re-assign their events to the new class, or shall disable event generation. The data object headers usually have a static data object group number even though just the events are being re-assigned. (The event reporting variation is not re-assigned; it is the events generated by the points whose indexes are specified in the static data object group headers that are being re-assigned to an event class.) An object header and index list as shown in Figure 12 and its accompanying description may be used in the requests if the outstation supports assignment of individual points.

The class object headers in an assignment set always use object group 60 (class data) and qualifier 06 (all). Variations 2, 3, and 4 correspond to class assignments of 1, 2, and 3, respectively. Variation 1 specifies that event generation shall be disabled; it does **not** mean that events are reported in requests for class 0 data! No other variations are permitted.

Table 12 shows which object header groups and variations are used for specifying the points whose events are to be re-assigned.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 12—Object headers used for re-assigning event classes

Point type or data type	Object headers in the assignment message		Applies to events reported using group
	Group	Variation	
Binary Input	1	0	2
Double-bit Input	3	0	4
Analog Input	30	0	32
Frozen Analog Input	31	0	33
Counter	20	0	22
Frozen Counter	21	0	23
Binary Output Status	10	0	11
Binary Output Command	12	0	13
Analog Output Status	40	0	42
Analog Output Command	41	0	43
File	70	0	70 (variations 4, 5, 6, & 7)
Octet String	110	0	111
Virtual Terminal	112	0	113

Events receive their class attribute at the time they are created. Thus, a point that is requested to assign its events to another class does not have to change already buffered events to the new class; only events generated after the ***assign class*** request are assigned to the new class.

The response to an ASSIGN_CLASS function code request is always a null response.

NOTE

Masters should issue a poll for all three event classes in a single request immediately after issuing an assign class request. This picks up any events that were generated between the most recent event poll, or unsolicited response, and the assign class request and assures that the events are reported in the same sequence that they were generated. If the poll for all three classes is not issued, there is the potential for events to be reported out of order, depending on the circumstances of which events were generated before and after the assign class request and the specific event polls issued by the master.

Outstation devices that are able to store the most recent class assignments in non-volatile memory may use those assignments after a device restart; otherwise, devices shall revert to their default class assignments.

4.4.14.1 Examples

EX 26	This example illustrates assignment of binary input events to class 1, analog input events to class 2, and frozen counter events for indexes 0 to 2 to class 2, and disabling of frozen counter events for indexes 3 to 10.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

►►► Request Message (beginning)

C3	16	3C	02	06	01	00	06	3C	03	06
AC	FC	Grp	Var	Qual	Grp	Var	Qual	Grp	Var	Qual
		←		Assignment set			→	←		

►►► Continuation of Request Message

1E	00	06	15	00	00	00	02	3C	01	06
Grp	Var	Qual	Grp	Var	Qual	Range		Grp	Var	Qual
		Assignment set					→	←		

►►► Continuation of Request Message

15	00	00	03	0A
Grp	Var	Qual	Range	
	Assignment set		→	

◄◄◄ Response Message

C3	81	00	00	
AC	FC	IIN_1	IIN_2	s

4.4.15 Function code 23 (0x17)

Delay Measurement

The master uses this function code to measure the **communication channel delay time**. It is most often used in the time synchronization process for non-LAN applications. The master shall measure delays that exist in the modems and communication media so that it can send a time value that shall be accurate when it arrives. For more information, see Clause 10.

The request message contains no objects.

The response message contains a single Fine Time Delay DNP3 object, g52v2. The Fine Time Delay object holds the outstation **processing delay,** which is valid for a single, one-time-only request and is defined as the number of milliseconds **from** the arrival of the leading edge of the start bit in the first data link octet of the received request message (generally at the interface where the bit is detectable from the physical media, such as at the output of the outstation modem if so equipped) **to** when the leading edge of the start bit of the first data link octet in the transmitted response message as it leaves the device (generally at the interface where the bit is placed onto the physical media, such as at the input of the outstation modem if so equipped). The processing delay includes clear-to-send timing.

The master shall record two times in order to compute the communication delay. The first is the exact instant when the leading edge of the start bit of the first octet in the transmitted request message leaves the master device (generally at the interface where the bit is placed onto the physical media, such as at the input of the master modem if so equipped. The time recorded is at the end of clear-to-send timing.) This time is identified as T_o (time out).

The second time that the master shall record is the exact instant when the leading edge of the start bit of the first octet in the received response message arrives at the master device (generally at the interface where the bit is detectable from the physical media, such as at the output of the master modem if so equipped.) This time is identified as T_i (time in).

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

The master calculates the communication channel delay time as follows[13]:

$$\text{Communication channel delay time} = (T_i - T_o - \text{processing delay}) / 2$$

NOTE

> The frequency of measuring the propagation delay is system dependent. Implementers need to consider whether the delay is fixed or variable. If variable, how rapidly does it change? The results of this analysis should indicate whether a propagation delay measurement is required only once, prior to every time synchronization message or at an interval somewhere between those extremes.

4.4.15.1 Rules

— **Rule 1:** Masters shall not retry **delay measurement** requests, and outstations shall not retry corresponding **response** messages at either the Application or the Data Link Layer. For more information, see Clause 10.

— **Rule 2:** Any outstation that sets IIN.4 [NEED_TIME] to request a time synchronization from the master shall support this function code.

— **Rule 3:** All masters that require time-stamped events shall support this function code.

4.4.15.2 Example

EX 27	This is a sample delay measurement where the processing delay is 18 milliseconds.

►►► Request Message

C3	17
AC	FC

◄◄◄ Response Message

C3	81	00	00	34	02	07	01	12	00
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Time Delay	

4.4.16 Function code 24 (0x18)

Record Current Time

This function code is used in the procedure for time synchronizing outstation devices that communicate over a LAN. It requests a receiver to record the time of receipt of the last octet in a message having this function code. Later, the receiver compares this time with the time sent by the master in a subsequent ***write*** message having a Last Recorded Time object, g50v3. The receiver uses these two pieces of information to correct its internal time reference.

[13] There are several assumptions implicit in this computation. First is that the outbound communication delay is the same as the inbound communication delay. The second is that the master and outstation Application Layers have a means of working together with the lower layers in order to obtain the times when start bits are received and transmitted. Third, the computed communication delay is applicable for the synchronizing message used to write the time; that is, the software design is such that other unpredictable processing delays (e.g., network delays and task switches) do not affect the accuracy.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A master that sends this message shall also record the time of transmission when it sends the last octet of a message with this function code. The master uses that time value in a subsequent message to the same outstation, having a Last Recorded Time object, g50v3.

For more information, see Clause 10.

The request message has no DNP3 objects, and the response message is a null response.

4.4.16.1 Rules

— **Rule 1:** Masters shall not retry **record current time** request messages at either the Application or the Data Link Layer. For more information, see Clause 10.

— **Rule 2:** Any outstation that has a TCP/IP interface and sets IIN.4 [NEED_TIME] to request a time synchronization from the master shall support this function code.

— **Rule 3:** All masters that require time-stamped events and support TCP/IP shall also support this function code.

4.4.16.2 Example

EX 28	This is an example of a master's request to record the current time and the outstation's response.

►►► Request Message

C3	18
AC	FC

◄◄◄ Response Message

C3	81	00	00
AC	FC	IIN_1	IIN_2

4.4.17 Function codes 25 (0x19), 26 (0x1A), 27 (0x1B), and 30 (0x1E)

Open File, Close File, Delete File, *and* Abort File

The purpose of the OPEN_FILE function code is to make a file available for reading or writing by the master and specifically for locking it so that no other process may access the file during this time. When the master is finished, it uses the CLOSE_FILE or ABORT_FILE function code to unlock the file, thereby making it available to another process. A master uses the DELETE_FILE function code to remove a file.

Open and ***delete*** are secure transactions. A valid authentication key (see 4.4.19) may be required to successfully perform these transactions and expires as soon as it is used. A zero value for the authentication key implies world (or guest) permissions.

Two DNP3 objects exist to support the open, close, delete, and abort functionality.

— A File Command object, g70v3.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— A File Command Status object, g70v4.

A File Command object is used to initiate open and delete operations.

A File Command Status object is used to indicate the success of ***open***, ***close***, ***delete***, and ***abort*** commands and to return a file handle during opens. It is also used to initiate file close and abort operations using a previously acquired file handle.

4.4.17.1 Preliminary notes

4.4.17.1.1 File handles

Most requests and all responses contain a file handle. This handle is a shorthand code for referencing a file using just four octets (32 bits). The outstation assigns file handles. File handle zero is reserved to indicate an invalid or illegal file handle; therefore, the outstation shall never assign this number to an opened file.

NOTE A master should never assume the same file handle is always associated with a particular file. Instead it should assume a different handle is assigned each time the file is opened or reopened.

If there is no activity referencing a file handle for a configurable length of time, the outstation shall automatically close the file. The timeout value shall be configurable up to one hour. When a file handle timeout condition occurs, the outstation shall send a File Transport Status event object, g70v6, using a status code value of *file handle expired* (17). The final state of the file closed under this condition is undefined.

4.4.17.1.2 File Command Status Objects

Since it may take some amount of time to determine the success of an operation, a File Command Status object, g70v4, is classified as an event object **when it is used in a response**. An instance of this DNP3 object is generated in every response to an open, close, delete, or abort request. Because they represent events, these objects may be configured for class 1, 2, or 3 and returned when class data is transmitted (polled or unsolicited). The master may also explicitly poll for these objects.

It is also acceptable for the outstation to return this object immediately in the response if the results are known without delay. If it is impractical to return the object immediately, the outstation shall instead immediately return a null response and then send the File Command Status object at a later time in an event report.

A File Command Status DNP3 object does **not** represent an event when used in a request.

NOTE Because a File Command Status object is an event object when returned in a response, the outstation should request an Application Layer confirmation in the response's application control octet.

4.4.17.1.3 File Transport Status objects

File Transport Status objects, g70v6, are used in the responses resulting from requests to write file data and in error responses resulting from requests to read file data. This object is also used to notify the master when the file status spontaneously changes such as happens when the open-file inactivity timer expires.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

This object is not used for open, close, delete, or abort requests. It is included in this portion of the document because of its similarity to a File Command Status object and because outstation files are automatically closed when conditions exist that necessitate a non-zero status code.

A File Transport Status object is classified as an event object. Because the data in these objects represent events, they may be returned when class data is transmitted (polled or unsolicited). The master may also explicitly poll for this data using these object headers in a ***read*** request.

It is also acceptable for the outstation to return this object immediately in the response if the results are known without delay. If it is impractical to return the object immediately, the outstation shall instead immediately return a null response and then send the File Transport Status object at a later time in an event report.

When the outstation detects an error and returns this object with a non-zero status code, the outstation shall also close the file, and the outstation is no longer required to retain information relative to when the file was opened. The final state of a file is undefined when it is closed by the conditions that create a File Transport Status event object with a non-zero status code value.

4.4.17.1.4 Additional information

Additional information about sequential file transfer is provided in Clause 5.

4.4.17.2 Opening a file

To open a file, the master issues an ***open*** command (OPEN_FILE function code) with the File Command object, g70v3. An ***open*** request directs the outstation to prepare to read or write the file and to assign a unique handle that shall serve as an alias for subsequent operations on the file.

4.4.17.2.1 Request messages

The master enters the *file name string offset*, *file name string size*, *permissions*, *authentication key*, *request ID*, and *file name string* octet fields into a File Command object. The *file name string* field shall be fully described as in

Path\FileName

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

If the file is to be **opened for reading**, the *operational mode* field is set to 0x01 and the *file size* and *time of creation* fields are set to zero. The *maximum block size* field is set to the maximum number of file octets the master can accept from the outstation in the response messages to each ***read*** request; the outstation may send less than this number. The virtual file position pointer in the outstation is set to zero,[14] which is the beginning of the file.

If the file is to be **opened for writing**, the *operational mode* field is set to 0x02 to **write a new file or to overwrite** an existing file. The *operational mode* field is set to 0x03 to **append octets** to the end of an existing file. The *time of creation* and *permissions* fields are set to appropriate values that the outstation shall use for attributes of the new file. The *maximum block size* field is set to the maximum number of octets that the master shall send in each ***write*** request message; the outstation may negotiate a smaller value in the corresponding field of its response. The *file size* field is entered by the master and specifies the total number of file octets that shall be transferred by all of the ***write*** requests combined, after the file is open. A file size of 0xFFFFFFFF indicates that the actual file size is unknown. Outstation devices are not required to accept unknown file sizes and may reject the request.

The *Request ID* in the request's File Command object is a master-dependent parameter that the outstation is required to return in the *request ID* field of its corresponding response. It permits the master to associate a response with a particular request. Outstations do not interpret or deduce any meaning from the value in this field.

When the outstation receives an ***open*** request with the operation mode set to 0x02, it shall truncate the file to a zero length, set the virtual file position pointer to zero, and set the time of creation and permission attributes per the respective fields in the File Command object. When the outstation receives an ***open*** request with the operation mode set to 0x03, it shall overwrite the time of creation and permission attributes per the respective fields in the File Command object, and it shall set the virtual file position pointer to one octet beyond the end of the file.

When an outstation is opened for writing and it receives an end-of-file indication during the transfer of the file contents (by virtue of the *Last* bit being set in the File Transfer object, g70v5), and the total number of octets transferred is less than the number specified by the *file size* field in this object, the outstation shall assume the file is completely received without error. However, if the total number of octets transferred is greater than the number specified in by the *file size* field in this object, the outstation shall consider this as an error.

4.4.17.2.2 Response messages

The outstation responds with a File Command Status DNP3 object, g70v4. See 4.4.17.1.2.

A status code of zero indicates that the file was successfully opened. The application should use the returned file handle in future references to the file until it is closed. A non-zero status code indicates that the file was not opened, and that the file handle value is invalid.

If the file was **opened for reading**, the *file handle* field contains the 32-bit handle to use in subsequent file request messages. If the *status code* field contains a non-zero value, indicating an error of some kind, then the file handle shall be set to zero. The *file size* field indicates the total number of octets in the file that was opened. The *maximum block size* field indicates the largest number of octets the outstation shall return with each ***read*** request; this value shall be less than or equal to the value in *maximum block size* field of the request. Note that the outstation may return fewer than this number in each ***read*** response.

[14] The virtual file position pointer is an indicator, private to and maintained by the outstation, for tracking the octet position within the file, relative to the beginning, from where the next octet should be read or to where the next octet should be written. Each time an octet is read or written, the virtual file position pointer is incremented by one count; otherwise it remains constant between requests. The virtual file position pointer can only be set from the master via an open request, and its value cannot be read by the master.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

If the file is to be **opened for writing**, the *file handle* field contains the 32-bit handle to use in subsequent file request messages. If the *status code* field contains a non-zero value, indicating an error of some kind, then the file handle shall be set to zero. The *file size* field should contain a zero. The *maximum block size* field indicates the largest number of octets the outstation can accept in each ***write*** request; this value shall be less than or equal to the value in *maximum block size* field of the request. Note that the master may write fewer than this number in each ***write*** request.

4.4.17.2.3 Examples

EX 29	This is an example of opening a file named "ABC" for reading where the outstation is unable to respond immediately. The block size specified is 192, and the request ID is 311. Assume that file events are configured for class 3.

►►► Request Message (beginning)

C3	19	46	03	5B	01	1D	00	1A	00	03
AC	FC	Grp	Var	Qual	Range	Size prefix		←		

►►► Continuation of Request Message

00	00	00	00	00	00	00	FF	01	00	00
File Command object										

►►► Continuation of Request Message

00	00	00	00	00	00	01	00	C0	00	37
File Command object continued										

►►► Continuation of Request Message

01	41	42	43
			→

◄◄◄ Response Message

C3	81	00	00
AC	FC	IIN_1	IIN_2

• • • N master polls requesting a quantity of two File Command Status objects. Each request is followed by an outstation null response. • • •

►►► Request Message

CE	01	46	04	07	02
AC	FC	Grp	Var	Qual	Range

◄◄◄ Response Message (beginning)

EE	81	00	00	46	04	5B	01	0D	00	66
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Size prefix		←

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

◄◄◄ Continuation of Response Message

77	88	99	00	08	00	00	C0	00	37	01
				File Command Status object						

◄◄◄ Continuation of Response Message

00	
→	|

►►► Confirm Message

	CE	00	
|	AC	| FC	|

NOTE

> The preceding example shows the master polling exclusively for File Command Status objects after requesting a file open. It is permissible, and even encouraged, for a master to interleave other requests during this time to more effectively utilize available bandwidth. It is not intended that other transactions be significantly delayed during file transfers. This philosophy holds for all file transfer activities whereby the master does not receive an immediate response.

EX 30	This is an example of opening a file named "ABC" for writing where the outstation is able to respond immediately. The master requests a block size of 2048 octets, but the outstation negotiates a size of 192.

►►► Request Message (beginning)

	C3	19	46	03	5B	01	1D	00	1A	00	03
|	AC	| FC	| Grp	| Var	| Qual	| Range	| Size prefix		| ←		

►►► Continuation of Request Message

00	80	4E	8F	1F	EB	00	FF	01	00	00
				File Command object						

►►► Continuation of Request Message

00	00	00	08	00	00	02	00	00	08	37
				File Command object continued						

►►► Continuation of Request Message

01	41	42	43
			→ |

◄◄◄ Response Message (beginning)

	E3	81	00	00	46	04	5B	01	0D	00	66
|	AC	| FC	| IIN_1	| IIN_2	| Grp	| Var	| Qual	| Range	| Size prefix		| ←

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

◄◄◄ Continuation of Response Message

77	88	99	00	08	00	00	C0	00	37	01
				File Command Status object						

◄◄◄ Continuation of Response Message

00	
→	|

►►► Confirm Message

	C3	00	
|	AC	| FC	|

4.4.17.3 Closing a file

To close a file, the master issues a ***close*** command (CLOSE_FILE function code) with a File Command Status object, g70v4. A close request directs the outstation to invalidate the file handle, and if the file had been opened for writing or appending, to store the contents of the file in its final destination. Closing the file also releases the file, making it available to other processes in the outstation.

In a request, the *file handle* field in the File Command Status object shall be set to the same handle returned during the open transaction. The *file size*, *maximum block size*, and *status* fields shall be cleared to zeros. Optional American Standard Code for Information Interchange (ASCII) characters can be included at the end of the File Command Status object for information, but outstations may ignore them; therefore, a master should not depend on those octets having any significance to the outstation.

The outstation responds with a File Command Status object, g70v4. See 4.4.17.1.2. In the response, the *file handle, file size, maximum block size*, and *optional ASCII characters* fields shall match those in the request.

A status code of zero indicates that the file was successfully closed. A non-zero status code indicates that an error was detected.

NOTE—If the master chooses to end the operation for any reason including a timeout, it should send either a close or an abort command to the outstation to release the file for further use.

4.4.17.3.1 Example

EX 31	This is an example of closing a file.

►►► Request Message (beginning)

	C3	1A	46	04	5B	01	0D	00	66	77	88
|	AC	| FC	| Grp	| Var	| Qual	| Range	| Size prefix		| ←		

►►► Continuation of Request Message

99	00	00	00	00	00	00	37	01	00	
				File Command Status object					→	|

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

◄◄◄ Response Message (beginning)

E3	81	00	00	46	04	5B	01	0D	00	66
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Size prefix		←

◄◄◄ Continuation of Response Message

77	88	99	00	00	00	00	00	00	37	01
File Command Status object										

◄◄◄ Continuation of Response Message

00
→

►►► Confirm Message

C3	00
AC	FC

4.4.17.4 Deleting a file

To delete a file, the master issues a ***delete*** command (DELETE_FILE function code) with a File Command object, g70v3. A ***delete*** request directs the outstation to remove the file from its file store and to remove references to the file from the applicable file directory.

The master enters the *file name offset*, *file name size*, *permissions*, *authentication key*, *request ID*, and *file name* octet fields into the File Command object with appropriate values. The *file name* field shall be fully described as in

Path\FileName

The master sets the *operation mode* field to 0x00 and the *file size, time of creation*, and *maximum block size* fields to zero.

The outstation responds with a File Command Status object, g70v4. See 4.4.17.1.2. In the response, the *file handle, file size*, and *maximum block size* fields shall be zero.

A status code of zero indicates that the file was successfully deleted. A non-zero status code indicates that an error was detected.

NOTE—If the outstation receives a ***delete*** command for a file that is open, it should **not** delete the file and should return a status code 4, File Locked.

4.4.17.4.1 Example

EX 32	This is an example of deleting a file.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

►►► Request Message (beginning)

	C3	1B	46	03	5B	01	1D	00	1A	00	03
|	AC |	FC |	Grp |	Var |	Qual |	Range |	Size prefix		|	←	

►►► Continuation of Request Message

00	00	00	00	00	00	00	FF	01	00	00
				File Command object						

►►► Continuation of Request Message

00	00	00	00	00	00	00	00	00	00	37
				File Command object continued						

►►► Continuation of Request Message

01	41	42	43
			→ |

◄◄◄ Response Message (beginning)

	E3	81	00	00	46	04	5B	01	0D	00	00
|	AC |	FC |	IIN_1 |	IIN_2 |	Grp |	Var |	Qual |	Range |	Size prefix	|	←

◄◄◄ Continuation of Response Message

00	00	00	00	00	00	00	00	00	37	01
				File Command Status object						

◄◄◄ Continuation of Response Message

00	
→ |	

►►► Confirm Message

	C3	00	
|	AC |	FC |	

4.4.17.5 Aborting a file transfer

To abort a file transfer, the master issues an ***abort*** command (ABORT_FILE function code) with a File Command Status object, g70v4. An abort request indicates to the outstation that it shall immediately terminate the current read or write operation and close the file, **without saving** the file if the file were opened for writing or appending.

In the request, the *file handle* field in the File Command Status object shall be set to the same handle returned during the open transaction. The *file size*, *maximum block size*, and *status* fields shall be cleared to zeros. Optional ASCII characters can be included at the end of the File Command Status object for information, but outstations may ignore them; therefore, a master shall not depend on those octets having any significance to the outstation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

The outstation responds with a File Command Status object, g70v4. See 4.4.17.1.2. In the response, the *file handle, file size, maximum block size*, and *optional ASCII characters* fields shall match those in the request. If the outstation cannot abort the operation, it shall return a status code 9, Cannot Abort.

A status code of zero indicates that the operation was aborted and the file was successfully closed. A non-zero status code indicates that an error was detected.

NOTE

> After receipt of an ***abort*** command, in order to completely recover an original file that was being written, a outstation device may need to double-buffer the file during the file transfer operations; i.e., it may need an intermediate buffer to hold the file contents until a close command is received; at which time, the file data is written to the final destination.

4.4.17.5.1 Example

EX 33	This is an example of aborting a file transfer.

►►► Request Message (beginning)

	C3	1E	46	04	5B	01	0D	00	66	77	88
|	AC	| FC	| Grp	| Var	| Qual	| Range	| Size prefix		| ←		

►►► Continuation of Request Message

99	00	00	00	00	00	00	37	01	00
			File Command Status object						→ |

◄◄◄ Response Message (beginning)

	E3	81	00	00	46	04	5B	01	0D	00	66
|	AC	| FC	| IIN_1	| IIN_2	| Grp	| Var	| Qual	| Range	| Size prefix		| ←

◄◄◄ Continuation of Response Message

77	88	99	00	00	00	00	00	00	37	01
				File Command Status object						

◄◄◄ Continuation of Response Message

00	
→	|

►►► Confirm Message

	C3	00	
|	AC	| FC	|

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.4.18 Function code 28 (0x1C)

Get File Information

The purpose of the GET_FILE_INFO function code is for the master to retrieve information about a file. The file type, file size, time of creation, and permissions are returned.

A master obtains the information by issuing a GET_FILE_INFO function code with a File Descriptor object, g70v7, in the request. The *filename offset*, *file name size*, *request ID*, and *file name octet* fields are filled in with appropriate values. The *file name* field shall be fully described as in

Path\FileName

The *file type*, *file size*, *time of creation*, and *permissions* fields in the request object are cleared to zero.

If the file exists, the outstation responds with a File Descriptor object, g70v7. This object is an event object when it is used in a response, but it is not an event object when used in a request. The *filename offset*, *file name size*, *request ID*, and *file name octet* fields are filled in with the same data as in the request, and the *file type*, *file size*, *time of creation*, and *permissions* fields are completed with their actual values.

If the file does not exist, the outstation responds with a File Command Status object, g70v4, having the status code set to 3, File Not Found.

If the file referenced is a directory file, the *file type* field shall be zero to indicate a directory file and the *file size* field shall indicate the number of entries in the sub-directory. To retrieve file information on all files in the directory, issue a file read on the directory file. See Clause 5 for more details.

4.4.18.1 Example

EX 34	This is an example of requesting information about a file. Assume that file events are configured for class 3.

►►► Request Message (beginning)

C3	1C	46	07	5B	01	17	00	14	00	03
AC	FC	Grp	Var	Qual	Range	Size prefix		←		

►►► Continuation of Request Message

00	01	00	00	00	00	00	00	00	00	00
				File Descriptor object						

►►► Continuation of Request Message

00	00	00	00	37	01	41	42	43
				File Descriptor object				→

Because the outstation cannot fetch the file data immediately, it sends a null response and waits for the master to poll for event data.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

◄◄◄ Response Message

C3	81	00	00
AC	FC	IIN_1	IIN_2

• • • N master polls for class 3 data, each followed by an outstation null response • • •

►►► Request Message

CE	01	3C	04	06
AC	FC	Grp	Var	Qual

◄◄◄ Response Message (beginning)

EE	81	00	00	46	07	5B	01	17	00	14
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Size prefix		←

◄◄◄ Continuation of Response Message

00	03	00	01	00	00	08	00	00	80	4E
				File Descriptor object						

◄◄◄ Continuation of Response Message

8F	1F	EB	00	FF	01	37	01	41	42	43
				File Descriptor object continued						→

►►► Confirm Message

CE	00
AC	FC

4.4.19 Function code 29 (0x1D)

Authenticate File

This function code is used to obtain an authentication key that may be needed to open or delete a file. When implemented, an authentication key provides a form of security that assures the requestor can provide a user name and password acceptable to the outstation.

Not all outstations implement or require an authentication key. For those that do, the master shall submit a request to obtain an authentication key, and if the outstation recognizes a valid request, the outstation shall issue a key. The authentication key is applicable to only a single transaction, and it is entered in the *authentication key* field of the g70v3 object used with an open or delete request.

The master requests an authentication key using an Authentication object, g70v2. The *user name offset*, *user name size*, *password offset*, *password size*, *user name octet*, and *password octet* fields shall contain appropriate values. The *authentication key* field is cleared to zero in the request.

The outstation responds with an Authentication object, g70v2. If the request is acceptable to the outstation, the *authentication key* field contains a unique value that may be issued in an open or delete request; otherwise, if unacceptable, a zero appears in the *authentication key* field. This implies world (or guest) permissions. The *user name offset*, *user name size*, *password offset*, and *password size* fields contain zeros, and there are no user name octets or password octets included in the response for security reasons.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.4.19.1 Example

EX 35	This is an example of an authentication request and response.

►►► Request Message (beginning)

C3	1D	46	02	5B	01	15	00	0C	00	05
AC	FC	Grp	Var	Qual	Range	Size prefix		←		

►►► Continuation of Request Message

00	11	00	05	00	00	00	00	00	53	43
Authentication object										

►►► Continuation of Request Message

41	44	41	5A	45	42	52	41
Authentication object continued							→

◄◄◄ Response Message (beginning)

C3	81	00	00	46	02	5B	01	0C	00	00
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Size prefix		←

◄◄◄ Continuation of Response Message

00	00	00	00	00	00	00	AA	BB	CC	DD
Authentication object										→

4.4.20 Function code 31 (0x1F)

Activate Configuration

This function instructs the outstation to begin using the configuration or executable code specified by the objects included in the request. Executable code is defined here to mean processor instructions that are used for normal operation of the outstation device. It should not be confused with starting or stopping special applications, which is the purpose of function codes 17 and 18, ***start application*** and ***stop application***. The executable code discussed here is often called firmware.

The configuration data or executable code to be activated by the use of this command shall be loaded or installed in the outstation before issuing a request containing this function code. Those configuration values or code are not in use yet, but they are available to use when the ***activate configuration*** request is received. The ***activate configuration*** request does not contain new configuration values or processor instructions but references which of the available configuration values or sets of configuration values or executable code is to be activated. After processing the request, the specified items shall become active in the outstation from that time onward until superseded by another ***activate configuration*** request or by some other implementation-dependent method of altering the configuration or code.

Upon executing a request with this function code, the configuration data or executable code specified by the objects included in the request may be preserved in some form of non-volatile storage if the device is capable and shall be the configuration data or processor instructions that the device uses immediately following the request's execution.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Objects that are suitable for inclusion into an ***activate configuration*** request are

— File Specification String, g70v8

— Octet String, g110

These objects shall specify which configuration or processor instructions to activate and shall not contain the actual configuration values or code. The configuration values or executable code being activated shall be available in the outstation prior to receipt of the ***activate configuration*** request.

Requests shall include at least one object and may include more than one. If more than one object is included, the objects may have differing group and variation numbers.

The contents of the objects may specify the complete outstation configuration or only a portion of the configuration and are vendor specific. Similarly, the contents of the objects may specify the entire outstation executable code or only specific code sections. The method by which the master station determines which objects to include in an ***activate configuration*** request, and the content of those objects, are matters beyond the scope of this specification.

When an outstation receives an ***activate configuration*** request, it shall check the fitness or suitability of the corresponding objects in the request and may perform checks on the configuration data or executable code (such as a CRC check) referenced by objects in the request prior to initiating the activation operation. After completing the checking, the outstation shall immediately return a Status of Requested Operation object, g91v1.

If no errors are detected, the outstation shall complete the transmission of the Status of Requested Operation object before initiating the activation. If any errors are detected during the checking, the outstation shall not activate any of the configurations or processor instructions, even if some of the objects or data referenced by the objects are valid.

The state of the IIN bits in the ***activate configuration*** response shall reflect the current conditions and not the anticipated conditions. The status in object g91v1 is an indicator of whether the outstation expects the ***activate configuration*** operation to succeed or fail, and it is determined prior to initiating the activation. The device shall not set an internal indications bit, such as IIN2.5 [CONFIG_CORRUPT], due to failing a check that would cause a non-zero status code. This is because the new configuration data is not active; it is pending. The device shall not set the IIN1.7 [DEVICE_RESTART] bit in anticipation of restarting; it shall wait until after it has restarted to set that bit. IIN bits may be set for other reasons.

A Status of Requested Operation object contains a delay time during which the outstation expects to be busy and shall not respond to requests. It is permissible for an outstation to restart during this period. The master is not obligated to honor this time and may send requests during the delay period; however, because the outstation may not reply, the master may conclude that the outstation is off-line. For some systems, this is a useful means for notifying operators that the outstation is not available.

If the outstation does restart because of performing an activate configuration operation, its first response shall have the IIN1.7 [DEVICE_RESTART] bit set. Some outstations may detect that the new configuration is corrupt during the activation operation, or soon thereafter. When this happens, then the outstation shall set the IIN2.5 [CONFIG_CORRUPT] bit.

The dashed points as follows describe a few examples that use this function code. Vendors may choose other schemes not described here.

— The master station writes analog deadband data to a temporary file in the outstation named AnalogDeadbands.tmp. Next, the master reads the file and compares its contents to what it sent in order to assure that the outstation's file is intact. The master then issues an **activate configuration**

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

request with one object (g70v8) that identifies file AnalogDeadbands.tmp. The outstation now saves the deadband information from the file to non-volatile memory and uses the new deadbands for detecting analog events.

Alternately, if the outstation is designed to configure itself by reading from a file (e.g., AnalogDB.dat), instead of storing the new deadbands in non-volatile memory upon receipt of the ***activate configuration*** request, the outstation copies the contents of file AnalogDeadbands.tmp to AnalogDB.dat.

— An outstation designed for water systems contains multiple configuration sets for controlling pump sequencing. The configuration sets are located in non-volatile storage when the device is manufactured; the sets are named "Pump Seq A," "Pump Seq B," and "Pump Seq C." One of these sets shall be chosen to determine the operating sequence. An octet string (g110v10) containing "Pump Seq B" is included in the **activate configuration** request. After the request is processed, the "Pump Seq B" configuration set is used.

— An outstation uses a data set with the name "Control duration time." It has three values: a default control duration time, a beginning index number, and an ending index number. The master places 250 milliseconds, beginning at index 24 and ending at index 26 into a data set object (g87v1), and writes that data set to the outstation, where it is saved in volatile memory. No other action is initiated by the outstation at that time. The master then issues an **activate configuration** request with one object (g110v21) giving the name of the data set "Control duration time." After the request is processed, the default on-times for control point at indexes 24, 25, and 26 are set to 250 milliseconds and saved in non-volatile storage.

— A vendor fixes a bug in his embedded processor code. The code is downloaded to the outstation as a block of data bytes in RAM. The master issues an **activate configuration** request with an object (g110v56) containing a privately defined ASCII command string "Store 120188 bytes at address 0x001374756 CRC 0x2D94AC8B." Upon receipt, the outstation calculates a 32-bit CRC of all 120 188 bytes at address 0x001374756 and compares the result to 0x2D94AC8B. If the CRC values compare, a g91v1 object with status 0 is sent back to the master and then the 120 188 bytes are written to flash memory and a reboot is issued. If the CRC values do not compare, the outstation returns status code 2 indicating a problem in the 120 188 bytes. The outstation does not set any IIN bits and continues on as usual as if it had never received the **activate configuration** request.

—

NOTE

Depending on the system design, an outstation setting the IIN2.5 [CONFIG_CORRUPT] bit may be sufficient reason to download a new configuration.

The IIN1.7 [DEVICE_RESTART] bit should not be the sole indicator for a master to download a new configuration. The reason for this is due to outstations that restart as a normal activity when processing an ***activate configuration*** request. If IIN1.7 were used alone, the master and outstation could get into an endless loop of restarting, downloading, and activating configuration.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 13— Example exchange to activate configuration

Figure 13 is an example exchange between the master and the outstation. This example shows the master checking that corruption of the file did not occur during its download. DNP3 does not require these steps. The example also shows the behavior when the outstation does, and when it does not, detect an error in the configuration requested to be activated.

4.4.21 Function code 32 (0x20)

Authentication Request

The master uses this function code when sending authentication requests to the outstation.

For more information, see Clause 7.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.4.22 Function code 33 (0x21)

Authentication Error

The master uses this function code to transmit error messages to the outstation.

For more information, see Clause 7.

4.4.23 Function code 129 (0x81)

Response

All response messages except for unsolicited response messages use function code RESPONSE. Whenever the master issues a request, regardless of what function code appears in the request, the response shall always use function code RESPONSE.

Most of the previous examples illustrate this.

4.4.24 Function code 130 (0x82)

Unsolicited Response

Unsolicited responses always use function code UNSOLICITED_ RESPONSE without regard to which DNP3 objects are included.

4.4.25 Function code 131 (0x83)

Authentication Response

The outstation uses this function code to issue authentication messages to the master.

For more information, see Clause 7.

4.5 Detailed IIN bit descriptions

This suclause provides details for each of the Internal Indication bits.

4.5.1 IIN1.0—All-Stations Message Received [ALL_STATIONS]

An outstation shall maintain a special state for each master with which it communicates. That state is arbitrarily named AllStationsMessageReceived.

The IIN1.0 bit indicates the status of this AllStationsMessageReceived state:

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

IIN1.0 = 1 indicates the state is set because a message was received with one of the all-station addresses.

IIN1.0 = 0 indicates the state is cleared per conditions in Table 13.

After receipt of a request containing one of the All-Stations addresses[15] (0xFFFD to 0xFFFF), the outstation shall set the AllStationsMessageReceived state. That state shall remain set until certain conditions are met. The conditions depend on the specific All-Stations address in the message and are described in Table 13.

Table 13—All-Stations addresses

All-Stations address	Address name	Conditions for clearing the AllStationsMessageReceived state
0xFFFD	ALLCALL_ DONT_ CONFIRM	The outstation shall clear the AllStationsMessageReceived state after sending the first response to the master. The outstation shall not request an Application Layer confirmation unless there are reasons to do so other than receipt of an All-Stations message. A master sends All-Stations messages to this address when it wants to minimize the communications traffic and reduce bandwidth requirements.
0xFFFE	ALLCALL_ SHALL_ CONFIRM	The outstation shall clear the AllStationsMessageReceived state only after receiving a master's Application Layer confirmation. The outstation shall request an Application Layer confirmation for all response messages while the AllStationsMessageReceived state is set. This confirmation may be combined with confirmation for other reasons. A master sends All-Stations messages to this address when it needs positive acknowledgment that the All-Stations message was received by the outstations.
0xFFFF	ALLCALL_ OPTIONAL_ CONFIRM	The outstation can determine whether or not to require Application Layer confirmation in order to clear the AllStationsMessageReceived state. It shall choose either the conditions specified for address 0xFFFD or the conditions specified for address 0xFFFE. This address provides backward compatibility with versions of this specification prior to 2.00 when conditions were specified less rigorously. A master sends All-Stations messages to this address if it or any of the outstations in the system were designed before DNP3 had multiple All-Stations addresses.

4.5.2 IIN1.1—Additional Class 1 Event Data Is Available [CLASS_1_EVENTS]

This bit indicates when the outstation has **any** class 1 events that have not yet been reported in the current response message. In a polled environment, this bit signals the master that it needs to send at least one request to retrieve the remaining class 1 events. Transmit this bit set when there are class 1 events existing that have not been included in the current response.

[15] Addresses appear in Data Link Layer frames. The design of the Data Link Layer and Application Layer provides a means to pass to the Application Layer the destination address in "All-Stations" messages so that the IIN1.0 bit and confirmations can be handled correctly.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.5.3 IIN1.2—Additional Class 2 Event Data Is Available [CLASS_2_EVENTS]

This bit indicates when the outstation has **any** class 2 events that have not yet been reported in the current response message. In a polled environment, this bit signals the master that it needs to send at least one request to retrieve the remaining class 2 events. Transmit this bit set when there are class 2 events existing that have not been included in the current response.

4.5.4 IIN1.3—Additional Class 3 Event Data Is Available [CLASS_3_EVENTS]

This bit indicates when the outstation has **any** class 3 events that have not yet been reported in the current response message. In a polled environment, this bit signals the master that it needs to send at least one request to retrieve the remaining class 3 events. Transmit this bit set when there are class 3 events existing that have not been included in the current response.

4.5.5 IIN1.4—Time Synchronization Required [NEED_TIME]

This bit is set when the outstation requires time synchronization from the master. It shall not set this bit if it does not require time synchronization. Reasons why an outstation would not require time from the master include

— It is synchronized from another source (such as a global positioning system [GPS] clock or another master).

— It never sends time-stamped data.

If the outstation sets this bit, it shall support both

— Time synchronization messages.

— Clearing the bit via a **write** request from the master specifying object g80v1.

If the outstation sets this bit, the master shall promptly perform a write time sequence. If timekeeping is not required in the system, the master should simply issue a ***write*** request to clear IIN1.4.

If the master clears this bit by issuing a ***write*** request to object g80v1, the outstation shall

— Assume a time synchronization message is not forthcoming.

— Wait before setting this bit again for at least the interval that it normally waits after time synchronization is received.

Masters may send time synchronization messages even if the outstation does not set the IIN1.4 bit.

If an outstation sets the bit, it shall leave the bit set until it is cleared by the master. The outstation may not clear the bit until the master issues a time synchronization message or writes a 0 to IIN1.4.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

NOTE

In the past, vendors have interpreted this bit to mean one of two choices, namely the outstation sets IIN1.4

a) Well before the timing reference drifts beyond its rated accuracy in order to give the master time to complete its higher priority tasks before sending the time.

b) After the outstation's internal clock has drifted beyond its rated accuracy. For this choice, it is assumed that the master performs time synchronization periodically without the need to set the IIN1.4 bit.

Masters need to know what interpretation is implemented in each outstation. If the master cannot be certain whether an outstation has implemented choice a), and the system permits sufficiently frequent synchronization to meet specification, it should use interpretation b) for that outstation.

Failure of an outstation to receive time synchronization shall not cause the device to inhibit any function; the only side-effect of not receiving time is that the time information it reports or utilizes may be inaccurate.

NOTE

Outstations that set the IIN1.4 bit at unreasonably short intervals might adversely impact system operation by requiring a disproportionate amount of the available bandwidth for non-data collection activities.

4.5.6 IIN1.5—Some Output Points Are In Local Mode [LOCAL_CONTROL]

This bit shall be set whenever at least one of the outstation's output points are in the local operation mode.

This indication is not intended to inhibit or prevent controls from the master; it is an advisory that controls might not succeed. It is the system implementer's responsibility to provide suitable lockouts that disable control operation from the master when points are in local mode.

If the master directs a control command to a point in local mode, the outstation shall return a status code indicating failure due to the point being in local mode.

NOTE

Masters may send control requests to outstations that set IIN1.5 because this bit does not necessarily mean that all points are disabled for operation from the master. For example, in a data concentrator application, only one IED may be in the local mode thereby requiring the data concentrator to set IIN1.5; however, control commands to the other IEDs should succeed.

NOTE

Outstations that have a local-remote switch or logic are responsible for preventing control actions on those points that it places in local mode.

4.5.7 IIN1.6—Device Trouble [DEVICE_TROUBLE]

This bit is set when an abnormal condition exists in the outstation. This specification does not enumerate abnormal conditions because they are device specific. They often result from hardware problems such as over-temperature or a subsystem failure. IIN1.6 should only be set while the condition exists and another IIN bit cannot indicate the condition.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.5.8 IIN1.7—Device Restart [DEVICE_RESTART]

This bit is set when an outstation restarts for any reason. The only permissible method to clear the bit is for the master to specifically write a 0 to it using object g80v1 and index 7.

NOTE The master should clear IIN1.7 bits as soon as possible after detecting the bit is set so that it can detect additional resets. The bit should be cleared prior to downloading new parameters like analog deadbands.

4.5.9 IIN2.0—Function Code Not Implemented [NO_FUNC_CODE_SUPPORT]

This bit shall be set in responses when the master request contains

— A function code that is not supported by the outstation.

— A function code that is not supported for objects of the type specified in the request.

An example of a function code that possibly would not be supported by an outstation is a delay measurement, DELAY_MEASURE, if the outstation does not require time synchronization.

An example of a function code that possibly is not supported for objects of the type specified in the request is a freeze command, IMMED_FREEZE, sent with counter objects in the message if the outstation does not support frozen counts.

4.5.10 IIN2.1—Object Unknown [OBJECT_UNKNOWN]

The outstation shall set this bit in responses according to the conditions in Table 14.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 14—Conditions for setting IIN2.1

Request from master contains	Set IIN2.1 bit if	Do not set IIN2.1 bit if
Non-event data objects	The outstation does not support the requested operation on objects in the request without regard to the indexes specified in the objects.	Outstation does support the requested operation on objects in the request, even if one or more of the objects in the request specifies an index that the outstation does not have.
Event data objects.	The outstation does not support events of the types specified in the request objects - AND - events relating to the types in the request objects are not defined for the outstation's particular DNP3 subset implementation level.	The outstation does support events of the types specified in the request objects, even though it does not have any at the time of the request* or has fewer events to report than the master requested - OR - the outstation does not support events of the types specified in the request objects, but events are defined for the outstation's particular DNP3 subset implementation level*. * In both cases where there are no events to report, the outstation returns a null response.

An example of an IIN2.1 error is when a master requests a direct operate of an analog output point when the outstation has no analog outputs.

Another example of this error is when a master includes a Pattern Control Block object, g12v2, in the request in order to energize a relay and the outstation only supports a CROB object, g12v1.

An example where the outstation shall **not** set the bit is when a master requests to read analog input index 99 and the outstation does not have a point with that index; the outstation shall instead set IIN2.2 [PARAMETER_ERROR].

4.5.11 IIN2.2—Parameter Error [PARAMETER_ERROR]

The reader requires a thorough understanding of the qualifier and range fields in object headers and what is meant by object prefixes in order to know when an outstation shall set this bit. Subclause 4.2.2.7.3 describes these in depth.

The outstation shall set this bit in responses when it supports the requested operation on objects in the request, but not all of the points in the index range or object index prefixes exist in the outstation. The bit is set even if some of the points are available. It is not set when the qualifier octet in the object header does not specify indexes in the range field or an object index prefix.

This bit shall also be set in responses whenever the outstation is unable to parse the Application Layer fragment because either

— The message was incorrectly formed.

— The request used a qualifier or other option that the outstation does not support.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

An example of this error is a master that requests a read of analog input points in the range of 0 to 49 and the outstation has only 32 analog inputs with index numbers 0 through 31.

If the request contains a ***read*** function code, the outstation response may return objects for points that are available and omit those for points that do not exist; however, this behavior is not mandatory.

4.5.12 IIN2.3 —Event Buffer Overflow [EVENT_BUFFER_OVERFLOW]

This bit indicates that an event buffer overflow condition exists in the outstation and that at least one unconfirmed event was lost because the event buffers did not have enough room to store the information. See Figure 14.

IIN2.3 shall be set in responses following the loss of at least one or more events due to overflow of the outstation's event buffers. After setting this bit, the overflow condition continues until the outstation has storage available in each of its event buffers to hold information for at least one more event.

It is not sufficient to report this bit as set in only a single message following the overflow. The outstation shall continue to set IIN2.3 in responses until the master has read and confirmed enough events so that when a new event of any type is generated, its information can be stored in the event buffers without causing an event loss.

Figure 14 illustrates setting and clearing of the IIN2.3 bit. It assumes an outstation device has an event buffer designed to hold 200 events. The figure shows the time relationships for the messages with the earliest time at the top of the diagram. The longer arrows pointing downward at an angle indicate messages in transit to the outstation or master. The rectangular box represents the number of events in the event buffer.

The hypothetical device in this example may have other event buffers, which for simplicity are empty.

NOTE

Note that the master in Figure 14 is poorly configured. It should have polled for event data more frequently in order to remove events before any were lost due to overflow.

In normal operation, the master should poll often enough to retrieve events before an overflow condition occurs. If the bit is set, it may indicate a problem in the system such as the master not polling frequently enough or possibly a high noise situation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 14—Event buffer overflow example

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.5.13 IIN2.4—Operation Is Already Executing [ALREADY_EXECUTING]

This bit is set when the requested operation was not performed or initiated because execution was in progress on the specified points, or applications, when the request arrived. Setting this bit is optional; outstations are not required to implement its functionality.

IIN2.4 bit applies to only the function codes in Table 15 as follows.

Table 15—Conditions for setting IIN2.4

Function code		Conditions when IIN2.4 could be set
3	SELECT	An operation is in progress that prevents a selection.
4	OPERATE	An operation is in progress that prevents the control action.
5	DIRECT_OPERATE	An operation is in progress that prevents the control action.
16	INITIALIZE_APPL	Application is already running.
17	START_APPL	Application is already running.
18	STOP_APPL	Application is not running.
19	SAVE_CONFIG	Configuration saving is currently in progress.
31	ACTIVATE_CONFIG	The application is currently using the configuration specified.

4.5.14 IIN2.5—Configuration Corrupt [CONFIG_CORRUPT]

This bit is set when the outstation detects a corrupt configuration that affects reliable functioning of the unit. It may also be set when other, non-critical configurations are corrupt. Setting this bit is optional; outstations are not required to implement its functionality.

4.5.15 IIN2.6—Reserved Bit [RESERVED_2]

This bit is reserved for future use. It shall always be returned cleared to 0.

4.5.16 IIN2.7—Reserved Bit [RESERVED_1]

This bit is reserved for future use. It shall always be returned cleared to 0.

NOTE

> In DNP3 documentation prior to version 2.00, IIN2.6 and IIN2.7 were described as being reserved for use by agreement. Because there was no assurance of interoperability among different vendors, these two bits are now reserved by DNP3 for future requirements.

4.6 Unsolicited responses

Unsolicited responses are messages spontaneously sent from an outstation without a specific request from a master when “something of significance” occurs.

The DNP3 protocol includes support for unsolicited responses. Masters and outstations are not obligated to implement unsolicited messaging as this feature is optional, but vendors that do implement this operation method may have a broader market in which to sell.

This method of operating has advantages in some applications. In a system with a large number of outstations and a single master, changes at an outstation can reach the master often much faster because

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

there is no delay while waiting for a master poll. The communication costs to achieve faster polling in some installations can be prohibitive, and the quickest notification of changes can occur if most of the messages contain only changes and confirmations. Unsolicited operation may reduce costs where the owners choose a "cost-per-byte" type of service.

On the other hand, equipment that implements unsolicited messages is more complex because the issues of media access and collision avoidance must be considered. Master software requires accepting messages from any of its outstations at any time. Another disadvantage is that system performance may become unpredictable during periods of heavy communication.

Employing unsolicited reporting requires an engineering judgment based on numerous factors for each individual system. There are no guarantees that unsolicited reporting is universally applicable for all systems.

It is not the purpose of this clause to justify using or not using unsolicited responses; rather it presents various requirements and considerations for those implementations that do support unsolicited operation.

4.6.1 Unsolicited response timing

Figure 15—Unsolicited timing diagram

Figure 15 illustrates two timing parameters associated with each transmitted unsolicited response.

a) First, is the time that it takes to transmit the response from the outstation to the master. This is shown as the *Unsolicited Message Transmit Time* in the figure.

b) The second timing value is the duration that the outstation waits to receive a confirmation back from the master. This is shown as the *Unsolicited Confirmation Timeout Period*. If the outstation sends an identical retry or a regenerated retry unsolicited response, it initiates transmission immediately upon confirmation timer timeout. This time does not need to be uniform from retry to retry.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.6.2 Outstation configuration

4.6.2.1 Compulsory configuration

Devices that support unsolicited responses shall support end-user configuration of the following parameters:

a) The *destination address* of the master device to send the unsolicited responses to.

b) The *unsolicited response mode* (either on or off). When unsolicited response operation is configured off, the device shall never send an unsolicited response, but otherwise responds to master requests.

c) The *timeout period for unsolicited response confirmation*. This is the amount of time that the outstation shall wait for an Application Layer confirmation back from the master. As a minimum, the range of configurable values shall include times from one second to one minute; however, devices may offer longer and shorter timeouts for systems with slower or faster media.

   This period may include a random component to minimize the chances of multiple outstations reporting at the same time when a common event initiates an unsolicited response series.

d) The *number of unsolicited retries*. This is the number of retries that an outstation transmits in each unsolicited response series if it does not receive confirmation back from the master. The configured value includes identical and regenerated retry messages. The unsolicited response series ends if confirmation is not received by the end of the unsolicited confirmation timeout period of the final retry. One of the choices shall provide for an indefinite (and potentially infinite) number of transmissions.

4.6.2.2 Non-compulsory configuration

Suggested additional, non-compulsory, configuration parameters are:

a) *Hold time before initiating an unsolicited response.* Delaying for a configurable amount of time after detecting each new event is often beneficial for allowing multiple changes to complete prior to transmitting an unsolicited response message. Vendors may choose whether the hold-time timer is retriggered for each new event detected (increased possibility of capturing all the changes in a single response), is not retriggered (giving the master a guaranteed update time), or is user selectable. A configured value of 0 indicates that responses are not delayed due to this parameter.

b) *Number of queued events before initiating an unsolicited response.* When events occur too often, and the hold time (previous parameter) is relatively long, the event buffer or queue may continue to accumulate events before an unsolicited message is transmitted. Without this counter, the buffer or queue can overflow causing a loss of events. With the counter, a message is initiated when its count reaches the configured amount. This gives time to transmit a quantity of events and then remove the acknowledged events from the queue, thus making room for more.

NOTE 1—Configuration parameters a) and b) are used together so that if either one of the criteria are met, an unsolicited response is transmitted. Either or both can be disabled by setting their values to 0.

NOTE 2—It is acceptable to have a separately configurable set of configuration parameters, 1 and 2, for each event class.

4.6.3 Support unsolicited enabling and disabling

Every master and outstation device that supports unsolicited responses shall also support function codes ENABLE_UNSOLICITED and DISABLE_UNSOLICITED. See 4.4.13 for details.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

As a minimum, an outstation shall accept commands to enable and disable unsolicited responses by event class (using object headers with group number 60), even if the device does not have class 1, 2, or 3 data when the request arrives. Enabling and disabled unsolicited messages on a per point type and index is optional.

4.6.4 Confirmation and application control octet

All unsolicited response messages shall request Application Layer confirmation, regardless of whether the response contains data or not (null response). This is accomplished by setting bit 5, CON, in the application control octet.

Outstations always set bit 4, UNS, in the application control octet of an unsolicited response. This indicates that the sequence number, SEQ, in the same octet is for an unsolicited response and not the sequence number for a solicited response.

Masters send Application Layer confirmations to unsolicited responses by setting the UNS bit in the application control octet and a sequence number, SEQ, matching that in the unsolicited response fragment being confirmed.

4.6.5 Device restart

Refer to the startup rules in Clause 5. These include considerations for unsolicited reporting.

4.6.6 Normal runtime behavior

Devices that support unsolicited responses shall obey the following rules:

> Unsolicited sequence numbers in the rules that follow refer to the sequence number that appears in the application control octet.
>
> Wherever the word confirmation is used, it refers to an Application Layer confirmation, function code CONFIRM. Data Link Layer confirmations, if used, never substitute for, replace, or imply Application Layer confirmations.

— **Rule 1:** An outstation shall only initiate unsolicited responses for those points that have been enabled.

NOTE

> Enabling and disabling points for initiating unsolicited responses is accomplished by sending the respective function code, ENABLE_UNSOLICITED or DISABLE_UNSOLICITED, with event class type objects (group 60), or if supported by the master and outstation, individual points may be specified using the appropriate static object group, variation 0, and the desired indexes.

— **Rule 2:** An outstation shall only include event data in an unsolicited response. Masters shall parse static data in unsolicited responses.

Previous versions of this specification[16] permitted sending static data for objects g10v2 and g40v2, for which no event objects existed. This behavior, while still permitted, is deprecated in version 2.00. Binary output status and analog output status event objects have been added to the Data Object Library, and their use is preferred.

[16] Sending static data objects for all analog, binary input, counter, and other point types in the initial unsolicited response at outstation startup was a part of the original specification. Early implementations with this behavior may still exist in the field.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— **Rule 3:** Unsolicited responses shall fit within a single fragment (FIR and FIN bits are both set in the application control octet). If the outstation has more data than fits within a single fragment, it shall include only as much data as fits into a single fragment. The outstation shall wait until that response is confirmed, and then it may create new unsolicited responses to transmit the remaining data.

— **Rule 4:** A master shall ignore the state of the FIR and FIN bits in the application control octet of a received unsolicited response. The reason for this rule is backward compatibility with outstations designed prior to version 2.00 of the DNP3 Specification that allowed sending of multiple fragment unsolicited responses.

— **Rule 5:** The SEQ number in the application control octet is advanced for original responses and regenerated retry messages and is unchanged for identical retries. The SEQ number in the outstation's startup null message (see Clause 5) is used as an initial unsolicited sequence number. Refer to 4.2.2.4.5.

— **Rule 6:** A master shall return confirmations to unsolicited responses immediately upon receipt, regardless of where it is in its polling sequence, even if it is waiting for a response to a solicited response.

— **Rule 7:** Masters shall examine the SEQ number in a received fragment and compare it with the previously received unsolicited fragment. If the SEQ numbers are different, it shall assume a new fragment has been received, but if the SEQ numbers are the same, it shall compare all of the octets following the IIN octets to determine if the response is new or a repeat.

— **Rule 8:** An outstation shall not initiate an original, an identical retry, or a regenerated retry unsolicited response while it is waiting for confirmation to a solicited response until either confirmation is received or the confirmation timer times out.

— **Rule 9:** An outstation shall not intentionally discard event information until it receives confirmation back that those events were received by the master.

— **Rule 10:** An outstation may only accept confirmations from the master if the application control octet contains a sequence number that matches the most recently transmitted unsolicited response. Once an outstation has advanced the unsolicited sequence number, late arriving confirmations with a previous sequence number shall be discarded.

— **Rule 11:** An outstation shall immediately clear corresponding event data from its event buffer(s) upon receipt of a confirmation during the confirmation timeout period shown in Figure 15. It shall perform the clearing action prior to responding to any pending requests or generating the next original unsolicited response. Receipt of a confirmation terminates the current unsolicited response series.

— **Rule 12:** If confirmation to an unsolicited response is not received before the confirmation timer times out, and the configured number of unsolicited response retries have not been sent, the outstation may choose to send an identical retry, or it may send a regenerated retry unsolicited response. If an identical retry is chosen, the sequence number shall be the same, and the retried fragment shall match the previous unsolicited response, octet-for-octet. If a regenerated retry unsolicited response is chosen, the outstation shall increment the unsolicited sequence number and may add new data or update data or modify the IIN octets from the previous unsolicited response.

— **Rule 13:** If the configured number of unsolicited response retries have been sent and confirmation is not received by the end of the final transmission's confirmation timeout period, the outstation shall terminate the current unsolicited response series, assume the confirmation is not forthcoming, and make the events available to be reported in a subsequent solicited response or a subsequent original unsolicited response. See Rule 14 for re-initiating unsolicited reporting.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— **Rule 14:** Implementers may choose an appropriate trigger to initiate a new unsolicited reporting series if confirmation is not received per Rule 13. The following alternatives may be used alone or in combination with each other:

1) A new event occurs.
2) The master issued to the outstation a ***read*** request for any data.
3) For connection-oriented systems, the master connected to the outstation and issued a request of any kind.
4) Wait for an extended time period after confirmation timeout of the final unsolicited response. This wait duration is usually much longer than the confirmation timeout period. The benefit of this option is that it does not depend on a stimulus from the field or from the master to begin again.

Implementers may choose other methods not listed here.

— **Rule 15:** If an outstation receives a request from the master having function code DISABLE_UNSOLICITED while it is waiting for a confirmation to an unsolicited response, regardless of which object headers appear in the disable request, it shall wait until either

1) Confirmation is received. At this time the outstation shall immediately clear the confirmed event data from its event buffer(s) according to Rule 11.
2) The confirmation timer times out. At this time the outstation shall terminate the current unsolicited response series[17] and cancel any expectation of confirmation for the entire unsolicited response. Make the events that were in the unsolicited response available for reporting in future responses.

If there are still any events available and enabled for reporting in unsolicited responses (because the request only disabled some but not all events), the outstation may initiate a new original unsolicited response at an appropriate time.

— **Rule 16:** If the outstation receives a **read** request of any kind while waiting for a confirmation to an unsolicited response, it shall defer building a response until either

1) The unsolicited response confirmation is received. The outstation shall then
   i) Immediately clear the confirmed event data from its event buffer(s) according to Rule 11.
   ii) Respond to the deferred read request as though it had just been received.
2) Timeout occurs waiting for the unsolicited confirmation. At this time, the outstation shall not send a retry unsolicited response; instead it
   i) Shall terminate the current unsolicited response series[17], assume the confirmation is not forthcoming, and make the events available to be reported in a subsequent solicited response or a subsequent original unsolicited response.
   ii) Respond to the deferred ***read*** request as though it had just been received and wait for confirmation to that ***read*** response if confirmation is required.
   iii) Thereafter, may formulate a new original unsolicited response for any remaining events that are permitted[18] to be transmitted in an unsolicited response.

NOTE—A timing diagram is provided in 4.6.7 illustrating the receipt of a ***read*** request while waiting for a confirmation to an unsolicited response.

[17] Termination of the unsolicited series does not apply to the initial null unsolicited response.
[18] Events are only permitted after the initial null unsolicited response has been confirmed and unsolicited responses are enabled as per the startup rules in Clause 5, which includes considerations for unsolicited reporting.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— **Rule 17:** If a non-read request of any kind is received during an unsolicited response series, the outstation shall respond immediately.

The non-read request does not cancel any pending expectation of unsolicited response confirmation.

If the response requires the outstation to set an IIN bit and request confirmation because, for example, a broadcast (all stations) message to address 0xFFFE was received immediately prior, the outstation shall

1) Set the IIN bit and request confirmation in the response and in future solicited and unsolicited responses until confirmation is received.
2) Clear the IIN bit, if appropriate, upon receipt of confirmation to a response in which the bit was set.

NOTE

> The outstation should coordinate the receipt of confirmations to unsolicited and solicited responses to assure that the IIN bit is only cleared if the IIN bit was set in the response being confirmed. For example, confirmation to an unsolicited response that did not have the IIN bit set does not imply the master received a solicited response with the IIN bit set, and vice versa.

— **Rule 18:** If a **read** response is deferred (see Rule 16), and another request of any kind is received from the master, then

1) If the new request is a ***read*** request, it replaces the already deferred request. It is deferred and handled according to Rule 16, as appropriate.
2) If the new request is a non-read request, the deferred read request is discarded and the new request is handled according to Rule 17.

4.6.7 Unsolicited message timing examples

This clause illustrates the behavior associated with unsolicited response messages.

Figure 16 shows the ideal situation where unsolicited responses are confirmed and there is no overlap between a master's request and the outstation's confirmation wait time.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 16—Ideal mixed unsolicited and solicited communications

Time advances from top to bottom in the diagram.

Each request, response, or confirmation is given a number inside square brackets [n] to aid in the description.

At the top, the outstation transmits an unsolicited response [1] and receives a confirmation [2]. The outstation removes or clears from its buffer(s) the events that it reported in the unsolicited response [1] because confirmation [2] was received from the master that the event(s) arrived there.

In the middle of the diagram, the master issues a read request [3] and the outstation responds [4]. The diagram assumes that the outstation required a read confirmation, and the master sends it [5]. The outstation removes or clears from its buffer(s) the events that it reported in the read response [4] because confirmation [5] was received from the master that the event(s) arrived there.

Later, the outstation transmits another new unsolicited response [6] and receives confirmation [7] from the master. The outstation again removes or clears from its buffer(s) the events that it reported in the unsolicited response [6] because confirmation [7] was received that the event(s) arrived at the master.

Everything worked as expected. Responses were transmitted and confirmations received.

In Figure 17, an example is shown where confirmations to unsolicited responses are not received as expected.

Figure 17 illustrates two retries of an unsolicited response before the unsolicited transaction is successful. At the top, the outstation sends an unsolicited response [1] that is not received by the master. The reason

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

does not matter; it could be a collision, noise burst, or something else. The outstation waits for the confirmation back.

The outstation waits for the confirmation timeout and then transmits an identical retry [2] of the original unsolicited response. This response contains the same Application Layer sequence number that the original used.

The confirm [3] from the master does not reach the outstation, but the outstation keeps waiting until its confirmation timer times out.

NOTE 1—The confirmation timeout durations for the original and retried unsolicited responses are not the same. In the diagram, the wait for a retry confirmation is much longer. The reader should only observe that the times are different but should not draw any conclusions regarding the relative time durations or which should be the longer. The specification allows implementers to vary the wait times according to their own strategy.

When retry confirmation times out, the outstation sends another identical retry [4]. This time confirmation [5] is received from the master, and the transaction is finally successful. The outstation shall now remove or clear from its buffer(s) the events that it reported in the unsolicited response [4] because confirmation [5] was received from the master.

NOTE 2—The master is able to determine that the second retry [4] is a repeat because the received message has the same sequence number and the message contents are the same as before [2]. In this case, the master ignores the processing of the contents of the repeated message [4] and just repeats the confirmation [5].

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 17—Unsolicited response or confirmation not received

Figure 18 illustrates two points—the outstation may regenerate unsolicited responses if new events occur and the outstation shall discard confirmations if the sequence number is not the same as the most recently transmitted.

The outstation sends an original unsolicited response [1], but the master is otherwise occupied and unable to send the confirmation in a timely manner. In the mean time, new events occur in the outstation. The confirmation [2] arrives at the outstation after the confirmation timer has timed out and after the outstation initiated another unsolicited response [3].

At confirmation timeout for unsolicited response [1], the outstation has the option of transmitting an identical retry of the original response or regenerated retry unsolicited response. The outstation chooses to send a regenerated retry unsolicited response [3] that includes the new events. Notice that the outstation incremented the sequence number. The master sends a confirmation [4].

In this example, the master almost certainly received duplicate events because the sequence numbers in unsolicited responses [1] and [3] were different, and the master is obligated to treat response [3] as a new message. Inasmuch as situations like this are possible, it is suggested that outstations transmit events with time stamps if the master requires that duplicated events are not processed. Time stamps provide information that a master can use to sort out the sequence of events and duplicates.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 18—Regenerated unsolicited response

Figure 19 illustrates a read request received while the outstation is waiting for a confirmation to an unsolicited response. At the top of the diagram, the original unsolicited response [1] is transmitted, but its confirmation [2] does not make it back to the outstation. The outstation waits until the confirmation timer times out and then initiates the sending of an identical retry [3]. However, the retried unsolicited response [3] does not reach the master.

By coincidence, the master issues a ***read*** request [4] that arrives at the outstation while the outstation is still waiting for the unsolicited response confirmation. The outstation defers the ***read*** request [4] and does not prepare a response. It continues to wait for a confirmation to its retried unsolicited response [3].

Because there is a deferred ***read*** request existing when the confirmation timer times out waiting for unsolicited response [3], the outstation makes all of the events that it transmitted in response [3] available for transmission in a solicited or unsolicited response.

Now the outstation transmits a response [5] to the deferred ***read*** request [4] and the master confirms [6].

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 19—Read request received while awaiting unsolicited confirm

Figure 20 illustrates the behavior when a non-read request arrives. As shown, the outstation sent an unsolicited response [1] that did not arrive intact at the master station. While the outstation waits for a confirmation, the master issues a ***select*** request [2] in order to perform a control operation. The outstation shall respond immediately to non-read requests, and it transmits a ***select*** response [3]. The master immediately follows up with an ***operate*** request [4], and the outstation initiates the control action and sends the operate response [5].

The control action caused new events. When the confirmation timer times out waiting for the confirmation to unsolicited response [1], the outstation chooses to regenerate an original unsolicited response [6] that contains the new events.

The confirmation [7] does not make it to the outstation and the confirmation timer times out once again. This time, there are no new events, and the outstation retries the unsolicited response [8]. Finally, confirmation [9] is received and the outstation clears the reported events from its buffer(s).

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 20—Non-read request received

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5. Application layer—part 2

5.1 Additional details

This clause contains additional implementation information and recommendations that are not found elsewhere in the document and provides further elaboration on topics already discussed.

There is no particular ordering to the presentation of topics.

5.1.1 Device startup

Whenever a master or outstation starts up after a reset for any reason, it shall perform certain activities to assure coordination with the device with which it communicates.

5.1.1.1 Outstation startup

The following rules apply when an outstation device restarts.

5.1.1.1.1 Outstation requirements

— **Rule 1:** An outstation device indicates that it has restarted by asserting the internal indications restart bit, IIN1.7 [DEVICE_RESTART] in its responses to a master poll and in unsolicited messages. It shall also set the IIN1.4 [NEED_TIME] if it needs the time.

— **Rule 2:** An outstation, configured to perform unsolicited reporting, shall immediately transmit an unsolicited null response message that has no data objects. If an Application Layer confirmation is not received within the configured timeout period, the outstation shall retry sending the unsolicited null response. It shall continue retrying, ignoring the number of configured unsolicited message retries (see Clause 4), until it receives an Application Layer confirmation from the master confirming the receipt of an unsolicited null response message.

— **Rule 3:** An outstation configured to perform unsolicited reporting shall not send data in an unsolicited response until it receives a request from the master containing function code ENABLE_UNSOLICITED that enables some or all points to initiate an unsolicited response. This does not mean the points do not generate events, only that the points cannot initiate unsolicited reporting of those events. The outstation shall ignore whatever unsolicited features were enabled prior to the restart; it may only apply those that are enabled after the restart.

— **Rule 4:** An outstation shall not discard events after the restart. It shall save the event data, even if it is configured to report unsolicited responses and it has not yet received an **enable unsolicited responses** message from the master to initiate unsolicited responses.

— **Rule 5:** An outstation, configured to perform unsolicited reporting, shall respond normally to any requests that it receives from a master.

NOTE | Outstation devices that are able to store the most recent class assignments from a master in non-volatile memory may use those assignments after a device restart; otherwise, devices should revert to their default class assignments.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5.1.1.1.2 Master requirements

— **Rule 1:** Upon detection of the IIN1.7 [DEVICE_RESTART] bit, the master shall issue a **write** command to clear this bit.

— **Rule 2:** If the IIN1.4 [NEED_TIME] bit was also set when the master detects the restart, it shall set the time in the outstation, subject to the considerations in Clause 4, according to the procedures specified in the time synchronization subclause of Clause 10.

— **Rule 3:** After clearing the outstation's IIN1.7 [DEVICE_RESTART] bit per Rule 1 and setting the time per Rule 2, the master shall next collect an initial data image from the outstation by issuing an integrity poll as described in 5.1.5.2.

— **Rule 4:** The master shall issue an initial set of class assignments if it desires to use the master assignments in lieu of those configured in the outstation.

— **Rule 5:** The master shall issue an initial set of analog deadbands if it desires to use the master assignments in lieu of those configured in the outstation.

— **Rule 6:** If the master supports unsolicited reporting and the outstation is configured to perform unsolicited reporting, the master shall enable unsolicited reporting in the outstation using the ENABLE_UNSOLICITED function code. This request shall only be issued after an integrity poll is complete.

5.1.1.2 Master startup

The following rules apply for **each** outstation supported when a master device restarts.

— **Rule 1:** If the master supports unsolicited reporting and the outstation is configured to perform unsolicited reporting, the master shall issue a message to disable unsolicited reporting of class 1, 2, and 3 events using the DISABLE_UNSOLICITED function code. This shall be the first request transmitted after the restart.

— **Rule 2:** The master shall collect an initial data image from the outstation device by issuing an integrity poll to it as described in 5.1.5.2.

— **Rule 3:** The master shall issue an initial set of class assignments if it desires to use the master assignments in lieu of those configured in the outstation.

— **Rule 4:** The master shall issue an initial set of analog deadbands if it desires to use the master assignments in lieu of those configured in the outstation.

— **Rule 5:** If the master supports unsolicited reporting and the outstation is configured to perform unsolicited reporting, the master shall enable unsolicited reporting in the outstation using the ENABLE_UNSOLICITED function code. This request shall only be issued after an integrity poll is complete.

5.1.2 Point index range recommendations

Point indexes serve to differentiate one point from another within the same point type. Experience has shown that assigning index numbers contiguously starting from zero results in transmissions that are more efficient. Assigning indexes in this manner also lowers the burden on some less sophisticated master software designs. Therefore, outstations should follow these recommendations:

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— Assign point indexes contiguously starting from index number zero for members of each point type.

— Within a point type, gaps in the point index range are permissible but should be avoided wherever possible. An example of an acceptable gap might be where a point is optional (i.e., where it may be installed or uninstalled without altering the point index values of the remainder of the points of the same point type). The assumption is that all points in the point index range from zero to the maximum installed point index always exist in the device, but some may be "absent" or currently not installed. Consider assigning the largest point indexes to the optional points so that if they are not installed, the remaining points still have a contiguous point index range.

— Assume that memory and database resources are required at the master for each index in the range of zero to the highest installed index number, irrespective of whether all these point indexes are actually transmitted.

— It is not acceptable to assign point indexes in a sparse manner (e.g., 0, 1, 2, 100, 101, 102, 200, 201, and 202). Point indexes are not intended, and should not be used, for grouping data or implying some relationship between data points.

— Consider assigning sets of optional points to one or more virtual devices, each with its own unique device address. When the points are installed, the corresponding virtual device exists, and if the points are not installed, the virtual device does not exist. This removes the possibility for introducing gaps in the point index ranges when points are installed or uninstalled.

— Consider providing a configurable mapping option that allows users to select which of the outstation's points are to be reported to, or controlled by, the master, and in what index order those points appear. In some situations, this makes it possible to keep the DNP3 point indexes contiguous beginning at 0. It also minimizes bandwidth requirements by not reporting points for which the user has no interest.

5.1.3 Event reporting

NOTE

> In general, the most bandwidth-efficient way to update a master is by frequently polling for events from the outstation and only occasionally requesting static data as an integrity measure. If the number of events generated between outstation interrogations is low, this allows the master to poll a larger number of outstations on a channel in a given amount of time. Each outstation returns much less data than if the master had to retrieve all data on every poll cycle.

DNP3 events are associated with something significant happening. Examples are state changes, values exceeding some threshold, an analog exceeding its deadband, snapshots of data taken at a particular time, transient phenomena, and newly available information.

NOTE

> It is highly recommended that all outstation devices, except for units having only a few points, perform report-by-exception processing internally and respond to class polls for event data. This makes possible efficient use of bandwidth.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5.1.3.1 Event reporting requirements

— Any device that generates more than one Data Link Layer frame in a response to class 0 poll shall support event reporting for

  1) Binary input points and double-bit input points.
  2) Analog input points.
  3) Counter input points.

— Events shall be reported in the order specified in 5.1.5.1. It is not necessary, but it is acceptable, to intermix binary events with analog events (and other events) to keep every reported event in time sequenced order.

5.1.4 Data types in class data responses

Class data responses are messages sent by the outstation resulting from a ***read*** request by the master that includes an object header specifying object group 60. Object group 60 is specified in the object header of class data requests only and never appears in the object header of any response. Data objects in class data responses are reported using the appropriate object headers for the data types being reported.

Users may not want to include the static or event data of all configured points in class data responses. For example, outstations may contain data that may be of little or no use to some users. Or perhaps bandwidth constraints make it necessary to exclude non-essential data. There may also be specific reasons why a vendor or user would want to implement a polling strategy that omits certain items from class data responses. For all these reasons, it may be desirable to exclude specific data types or points from class data responses. Therefore, all devices except for very simple, low point-count devices shall be configurable as to which points to include in class data responses. This is achieved by assigning individual points to one of the four classes (static class 0 or event classes 1, 2, or 3) during configuration of the device. Some devices also accept commands from a master to assign points to specific classes.

In response to a class data request, outstations shall not report static or event data for points that are not assigned to one of the four classes. Masters can obtain the static data of such points (not assigned to one of the four classes) by issuing a ***read*** request containing the respective group number and point index in the object header (e.g., specific analog inputs are accessed by specifying group 30 and appropriate indices in the object header), or by issuing a ***read*** containing the respective group number and specifying all points of that data type (e.g., all analog inputs are accessed by specifying group 30 and qualifier 06).

NOTE | It is strongly recommended that the included points be assigned the lowest index numbers, starting with zero, and all of the excluded points use higher index numbers.

For counters and analog point types, it is permissible to report both frozen and non-frozen values in the same response if the master is able to handle both; however, devices that are able to report both shall provide configuration to disable this feature and report either the non-frozen or the frozen variations but not both.

5.1.4.1 Static data, group 60, variation 1

Class 0 data responses are messages sent by the outstation resulting from a ***read*** request by the master that includes an object header specifying object group 60, variation 1. Object group 60 shall not be used in the responses; the responses shall instead include the appropriate static data object(s) (Table 16).

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 16—Static data included in class 0 data response

Data type	Static data included in class 0 data response?
Binary Inputs (Group 1) Double-Bit Binary Inputs (Group 3) Binary Output Status (Group 10) Counters and Frozen Counters (Groups 20 and 21) Analog Inputs and Frozen Analog Inputs (Groups 30 and 31) Analog Output Status (Group 40)	All devices (except for very simple, low point-count devices) shall allow users to assign points to no class, or to one of the four classes (static class 0 or event classes 1, 2, or 3), by changing the device's configuration. The device shall then include the static data of all points assigned to any of the four classes in its Class 0 data response.
Data Sets (Group 87) Binary-Coded Decimal (Group 101) Unsigned Integer—8-bit (Group 102) Octet Strings (Group 110)	All devices that support these static data types shall be configurable as whether any or all points of the specific data type are included in the Class 0 data responses.
Analog Input Reporting Deadbands (Group 34) Internal Indications (Group 80) Virtual Terminal (Group 112)	Points of these static data types are never included in Class 0 data responses.

5.1.4.2 Event data, group 60, variations 2, 3, and 4

Class 1, 2, and/or 3 data responses (events class responses) are messages sent by the outstation resulting from a ***read*** request by the master that includes object header(s) specifying object group 60, variation 2, 3, or 4. Object group 60 shall not be used in the responses; the responses shall instead include the appropriate event data object(s). Responses may also include Common Time of Occurrence objects (Group 51) if the event data objects are reporting relative time (Table 17).

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 17—Event data included in events class responses

Data type	Event data included in events class responses?
Binary Inputs (Group 1) Double-Bit Binary Inputs (Group 3) Binary Output Status (Group 10) Counters and Frozen Counters (Groups 20 and 21) Analog Inputs and Frozen Analog Inputs (Groups 30 and 31) Analog Output Status (Group 40)	All devices (except for very simple, low point-count devices) shall allow users to assign points to no class, or to one of the four classes (static class 0 or event classes 1, 2, or 3), by changing the device's configuration. The device shall then include the appropriate event data for all points assigned to event classes 1, 2, or 3 in its events class responses.
Data Sets (Group 87) Octet Strings (Group 110) Virtual Terminal (Group 112)	All devices that support these static data types shall be configurable as whether appropriate event data of any or all points of the specific data type are included in events class responses.
File Transfer Objects (Group 70) Security Authentication (Group 120)	All devices that support these data types shall include the appropriate event data in events class responses.
Analog Input Reporting Deadbands (Group 34) Internal Indications (Group 80) Binary-Coded Decimal (Group 101) Unsigned Integer—8-bit (Group 102)	Events are never generated for points of these static data types.

5.1.5 Data processing order

5.1.5.1 Event and static data

5.1.5.1.1 Reporting binary input and double-bit input events

When an outstation formulates a DNP3 response containing binary input and/or double-bit event data, the binary input and double-bit input events in the response shall be returned in event time-of-occurrence sequence within the fragment regardless of their data type or point index (i.e., oldest events first, and newest events last). This rule applies irrespective of whether the event time is reported.

As an example, Table 18 illustrates eight buffered events and the order in which those events shall be reported. (For simplicity, only minutes and seconds are shown for time.)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 18—Example of event buffering and reporting order

Internal buffered ordering				Response fragment ordering			
Data type	Point index	Time	Default variation	Data type	Point index	Time	Default variation
BI	25	4:17	1	DBI	10	2:34	2
DBI	12	2:51	2	BI	14	—	1
BI	14	2:37	1	BI	25	—	1
BI	25	2:55	1	BI	2	2:50	2
DBI	10	2:34	2	DBI	12	2:51	2
BI	2	2:50	2	BI	25	—	1
BI	25	2:38	1	BI	14	—	1
BI	14	3:05	1	BI	25	—	1

The four left-hand columns in Table 18 illustrate the hypothetical ordering of events within an event buffer and show the configured default variation to be used in responses when the master does not specify a reporting variation in its request.

The four right-hand columns show the ordering of events in the response to a poll for class 1, 2, and 3 data. The event at the top of the table appears in the fragment first, and the event at the bottom of the table appears last.

The example shows another aspect of reporting—events shall be reported in time sequence order even if, as Table 18 shows, it is necessary to create multiple object headers for the same object group and variation.

5.1.5.1.2 Reporting non-binary input and non-double-bit input events

When a DNP3 response contains counter, analog, or other event data where more than one event is present for an individual point, then all events for an individual point shall be returned in event time-of-occurrence sequence, earliest first. It is not necessary, as it is with binary and double-bit inputs, for the outstation to examine all the point indexes before constructing a response. In responses to class 1, 2, and 3 polls, there are no requirements for ordering data by point type, and events from the different point types may be inter-mixed or grouped together.

5.1.5.1.3 Mixed event and static data

If a DNP3 response contains event data and static data, the event data shall be reported before the static data. To allow for older outstations that build a response in the order of the object headers in the master's ***read*** request, master requests shall be formulated such that event object headers precede static object headers.

A master shall process DNP3 objects returned in a response in the same order that they appear in the received fragment. This causes the sequence of changes appearing in the master's database to correspond to the sequence order observed in the field, and the final state of each database point contains the most recently occurring event state. If a response contains event data and static data and there has not been an event buffer overflow, the final state after processing the event data should be the same as the static data state.

To update a master database in the same order as the corresponding field changes, it is necessary to precede static data reports with unreported event data for the same data points. The correct way to make this happen

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

is for the master to make a single ***read*** request for the required event and static data, and for the outstation to respond with any events preceding the static data in the response fragments. This applies to master requests for class data or for data from specific indexes.

NOTE Masters should count changes, set operator alarms, and log new states based on changes to the data in their database and not on whether the information was conveyed from the outstations via event or static objects.

Whenever a master restarts, an outstation device restarts or an outstation's event buffers overflow, the potential exists for losing data. To recover from this condition, masters should issue an integrity poll to the outstation, in order to collect all pending events and the present values of all points assigned to any of the four classes (static class 0 or event classes 1, 2, or 3). At the completion of this processing, the master will have an up-to-date image of the states of all points assigned to any of the four classes in the outstation. The master can then proceed to collect just events from the outstation and update its image of the outstation data with the reported events.

5.1.5.2 Integrity poll

A master should issue an integrity poll immediately after it detects that an outstation restarted (see 5.1.1.1) or after it restarts (see 5.1.1.2), and occasionally thereafter, for data reliability purposes, just to make sure that the master's database matches what is in the outstation.

An integrity poll requests all event data, followed by the static data of all points assigned to one of the four classes (static class 0 or event classes 1, 2, or 3). The recommended sequence of objects in an integrity poll is class 1, class 2, class 3, and class 0 data. In object group and variation terms, this is a single ***read*** request containing the following object headers in the order listed as follows:

a) Group 60, variation 2, qualifier 6

b) Group 60, variation 3, qualifier 6

c) Group 60, variation 4, qualifier 6

d) Group 60, variation 1, qualifier 6

The outstation responds with all event data, followed by the static data of all points assigned to one of the four classes, in one or more message fragments. If a point is not assigned to one of the four classes, its static data shall not be included in the response. If a response fragment contains events, then the outstation shall request an Application Layer confirm for that fragment.

5.1.5.3 Outstation event buffer overflow processing

An outstation device indicates that it experienced an event buffer overflow by asserting the event buffer overflow bit, IIN2.3 [EVENT_BUFFER_OVERFLOW], in responses to the master. When a master detects this situation, it should issue an integrity poll in order to reestablish the current state of all data in the outstation device.

5.1.6 Services provided

The Application Layer provides specific services to the layer above it.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5.1.6.1 Masters

The Application Layer provides the following services for the DNP3 User Layer in a master:

— Formats requests directed to one or more outstations.

— Notifies the DNP3 User Layer when new data or information arrives from an outstation.

5.1.6.2 Outstations

The Application Layer provides the following services for the DNP3 User Layer in an outstation:

— Notifies the DNP3 User Layer when action requests, such as control output, analog output, freeze and file operations, arrive from a master.

— Requests data and information from the outstation that is wanted by a master and formats the responses returned to a master.

— Assures that event data is successfully conveyed to a master (using Application Layer confirmation).

— Sends notifications to the master when the outstation restarts, has queued events, and requires time synchronization.

5.1.7 Services required

The Application Layer requires specific services from the layers beneath it.

— Partitioning of fragments into smaller portions for transport reliability.

— Knowledge of which device(s) were the source of received messages.

— Transmission of messages to specific devices or to all devices.

— Message integrity (i.e., error-free reception and transmission of messages).

— Knowledge of the time when messages arrive.

— Either precise times of transmission or the ability to set time values into outgoing messages.

5.2 Using virtual terminal objects

5.2.1 General

Many IEDs have separate, non-DNP3, serial interfaces used for configuration, diagnostics, and other ancillary tasks. These utility ports are often set up for connection to a dumb terminal or PC. The protocol on these ports is frequently an ASCII command set customized for the particular IED. Virtual terminal (VT) objects provide a somewhat transparent means to communicate with these ports by sharing the existing DNP3 bandwidth. This scheme eliminates the need for a parallel set of communication links.

Communications between the terminal equipment and processes at each end are transported over what is called a virtual communication channel (Figure 21). Each virtual communication channel is assigned a virtual port number.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 21—Virtual channels illustrated

The IED can be viewed as containing one or more terminal command interpreters. Each terminal command interpreter may be associated with a physical port or a virtual port, or both, at the discretion of the implementation. A key issue is that the IED treats the DNP3 virtual terminal channel just like it would a physical channel. If the IED supports more than one kind of command interpreter, unique DNP3 virtual terminal indexes are used to identify them.

5.2.2 Group 112 and 113 objects

VT objects emulate binary octet data streams going to and from a device. The Virtual Terminal Output Block, group 112, is used for data going to an outstation device, and the Virtual Terminal Event Data, group 113, is used for data coming from an outstation device. More details are provided in Annex A.

The procedure for communicating with these DNP3 objects is as follows. Master devices transmit data to outstation devices by writing one or more group 112 objects using the DNP3 point index number to specify the virtual port number.

Outstations return information to the master by responding to ***read*** requests for object group 113, responding to ***read*** requests for the configured event class, or transmitting an unsolicited response message.

The content of the data in the VT objects is specific to the particular terminal protocol. DNP3 masters and outstations do not need to know any of the details of the terminal protocol; DNP3 does not assign any significance to any of the data octets in VT objects.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5.2.3 Virtual terminal example

EX 36	In the following hypothetical example of a VT session: Data octets in the group 112 and 113 objects follow "data=" between pairs of single quotes ' '. The single quotes are not in the messages. <CR> represents the Carriage-Return character (0x0D). The point index is not shown but is assumed to be the same in all messages. Comments begin with a double forward slash, //, and continue to the end of the line. Comments are to clarify the example and do not appear in the message.

1) Master: Write g112v1, data='<CR>' // wakeup command
2) Outstation: Null response
3) Master: Read g113v0
4) Outstation: Null response // outstation has no data to send to master
5) Master: Read g113v0
6) Outstation: Respond with g113v3, data='OK<CR>'
7) Master: Write g112v6, data='CLEAR<CR>' // clear outstation command
8) Master: Write g112v7, data='LOGOFF<CR>' // end the IED session
9) Master: Read g113v0
10) Outstation: Respond with g113v7, data='OK<CR>BYE<CR>'

5.2.4 Discontinuous octet streams

Data flowing in either direction can be pictured as "discontinuous octet streams" whereby the gaps between characters are unpredictable—some long and some short. A set of octets in the stream is "packetized" into group 112 or 113 DNP3 objects; however, the packets do not necessarily delimit complete commands or responses. In the preceding example, packets do contain complete commands and responses, but that is not necessary.

Lines 7 and 8 show that the master is permitted to send multiple messages without intervening polls for responses.

Line 10 shows that the outstation accumulated and concatenated into a single event object the responses to the two ***write*** commands from lines 7 and 8. It would have also been acceptable for the outstation to have prepared two separate events and to have returned both event objects in the same message.

Masters are permitted to send partial or multiple commands within a single ***write*** operation. Similarly, an outstation may respond with partial information, and the master shall continue to read the data until there is no more.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5.2.5 Rules

DNP3 does not specify any explicit rules for how masters and outstations cooperate with each other in the issuing of terminal commands and responses. The rules are a local issue that depend on the terminal design. It is incumbent on the master and outstation user software to fully understand the command structure and the nature of the data streams between themselves.

Messages can flow in either direction at any time. There are no DNP3 procedures for the initiation or conclusion of a VT session; i.e., implicit connections exist by the mere presence of a VT compatible master and outstation.

The master and outstation split the responsibility for VT maintenance. Masters are responsible for maintaining an environment that allows outstations to return responses in a timely manner. In a polled system, the master shall periodically poll for outstation responses whenever a response is possible. In an unsolicited response environment, both ends shall check that the background data traffic is at a low enough level that outstation responses can be sent without impacting more important event data.

5.2.6 Virtual terminal bandwidth considerations

IEDs with large amounts of data transportable over virtual communication channels should be careful to limit their usage of the DNP3 link. One possibility is providing a configuration to select which class, 1, 2, or 3, that group 113 objects are assigned.

Another possibility is to limit the amount of virtual terminal data that outstations transmit to less than the allowed 255 maximum octets per message. Limiting the length of message strings ensures the greatest compatibility with memory-limited devices and allows other higher priority messages to be interspersed with VT messages. Choosing a data size so that responses fit within a single Data Link frame can sometimes simplify and reduce the amount of virtual terminal traffic.

In an unsolicited environment, it may be desirable for outstations to implement a scheme that constrains the amount of data they transmit (combination of message size and message frequency) in order to limit the overall system bandwidth utilized for this function.

5.3 Sequential file transfer

Sequential file transfer implies that complete files are read or written, starting from beginning to the end.

File transfer objects defined for group 70 variations 2 through 7 are applicable for sequential file transfer. The original File Identifier object g70v1 has been superseded.

Clause 4 describes the function codes associated with file transfer in detail, which are not repeated here. This clause is intended to explain how to use the functions and objects for reading and writing files and for determining which files exist in an outstation.

5.3.1 Authentication

An outstation may require authentication before opening or deleting a file. This requires the master to pass request authentication by issuing a request using function code AUTHENTICATE_FILE and including an Authentication object, g70v2. This request contains a user name and password.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

The outstation returns an Authentication object, g70v2, with an authentication key that is subsequently entered in the *authentication key* field of the g70v3 object appearing in the following open or delete request. The authentication key is valid for one use only. It expires as soon as it is used, or upon timeout at the outstation, whichever occurs first.

Outstations should generate keys using a private pseudo random algorithm; each request for an authentication key should yield a unique key. An authentication key value of 0 indicates that permission was denied or that the requested permission was not granted.

It is possible for there to be a set of user names and passwords that have different permissions, and outstations may use authentications to limit file access as is deemed suitable for the application. The details are a local issue beyond the scope of this specification.

5.3.2 File permissions

The creator of a file sets its permissions. Unless otherwise agreed to by the capabilities of the master and outstation, the bits in all three classifications, world, group, and owner, shall be set the same. Permissions when returned in any response shall indicate the current requestor's privileges, and the bits in all three classifications shall be set the same.

The **read** privilege gives the user the right to

— Read the file using function code OPEN_FILE with operation mode READ.

— Retrieve file information using function code GET_FILE_INFO.

The **write** privilege gives the user the right to

— Write the file using function code OPEN_FILE with operation modes WRITE and APPEND.

— Delete a file using function code DELETE_FILE.

The **execute** privileges gives the user the right to

— Initiate actions that use the file's contents when using a function code that includes a file specifier.

5.3.3 Reading a file

This clause provides an overview of the procedures for reading a file. The reader should refer to the function code descriptions in Clause 4 for details of each operation. Reading a file consists of the following transactions:

— Optional authentication.

— Open file.

— One or more read requests.

— Close file.

If the system requires an authentication key to open a file, the master issues an ***authentication*** request as outlined in 5.3.1.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Open file, ***read*****,** and ***close file*** requests return event objects. If the outstation is able to respond immediately, it shall return the appropriate event object. But if it cannot, the outstation responds immediately with a null response having no data objects, and thereafter it is the master's responsibility to poll for events (assuming unsolicited reporting is not used). The master may issue polls either for event classes or for the specific event object expected, depending on system configuration, until the expected response object is returned, or a local timer in the master times out.

NOTE

System designers need to coordinate which polls the master should issue while waiting for event file objects in the response. The outstation needs configuration for whether to return file event objects in polls for class event data. See Clause 4. The master should accommodate whichever way the outstation is configured.

It is permissible to use unsolicited reporting to return file event objects if the master and outstation devices support it and the system designer chooses this alternative.

To open a file, the master issues a message using function code OPEN_FILE. This message requires File Command object, g70v3, containing the appropriate information. See Clause 4 for details. A File Command Status event object, g70v4, is returned. If the status code in the returned object is zero, the file was successfully opened, and the application shall use the returned 32-bit file handle during ***read*** requests to the file until it is closed. A non-zero status code indicates that the file was not opened,and that the file handle value is invalid.

The maximum block size permitted for transferring file data is negotiated during the open process. For reading a file, the master states the maximum number of octets it can accept in the File Command object, g70v3. The outstation shall set a size, which is the lesser of the value in the master request or a limitation in the outstation, into the *maximum block size* field in the response's File Command Status object, g70v4. The outstation shall not transfer more octets than the block size it returned in the response.

To obtain the file data, the master issues one or more ***read*** requests using a File Transport object, g70v5, specifying the file handle and the block number. The block number starts at 0 in the first ***read*** request and shall increment by 1 for each subsequent ***read*** request. If no errors are detected, the outstation shall respond with a File Transport event object, g70v5, containing data read from the file.

NOTE

The master should not assume that each block, except the last, has an identical size. Both the outstation and the master should update their local file offset indicators after each ***read*** request.

NOTE—The master does not set the *last* bit in ***read*** requests, but the outstation does set it in the responses. See the description of a File Transport object, g70v5, in the *Data Object Library*.

If the outstation detects an error associated with the ***read*** request, it shall return a File Transport Status event object, g70v6, with an appropriate status code.

After all of the file data has been transferred from the outstation, the *last* bit being set in the response object, the master issues a ***close file*** request having a File Command Status object, g70v4. This object is not an event object when used in the request. The outstation responds with a message containing an identical File Command Status object as that which appeared in the request.

EX 37	This is an example of a file ***read*** request. Assume that file events are configured for class 3.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

►►► Request Message (beginning)

C3	01	46	05	5B	01	08	00	66	77	88
AC	FC	Grp	Var	Qual	Range	Size prefix		←		

►►► Continuation of Request Message

99	02	00	00	00
File Transport Status object				→

Because the outstation cannot fetch the file data immediately, it sends a null response and waits for the master to poll for event data.

◄◄◄ Response Message

C3	81	00	00
AC	FC	IIN_1	IIN_2

• • •

N master polls for class 3 data, each followed by an outstation null response.

• • •

►►► Request Message

CE	01	3C	04	06
AC	FC	Grp	Var	Qual

◄◄◄ Response Message (beginning)

EE	81	00	00	46	05	5B	01	16	00	66
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Size prefix		←

◄◄◄ Continuation of Response Message

77	88	99	00	00	00	80	A5	A5	A5	A5
File Transport Status object										

◄◄◄ Continuation of Response Message

A5	A5	A5	A5	A5	A5	A5	A5	A5	A5
File Transport Status object continued									→

►►► Confirm Message

CE	00
AC	FC

5.3.4 Writing a file

This clause provides an overview of the procedures for writing a file. The reader should refer to the function code descriptions in Clause 4 for details of each operation. Writing a file consists of the following transactions:

— Optional authentication.

— Open file.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— One or more write requests.

— Close file.

If the system requires an authentication key to open a file, the master issues an ***authentication*** request as outlined in 5.3.1.

Open file, ***write***, and ***close file*** requests return event objects. If the outstation is able to respond immediately, it shall return the appropriate event object. But if it cannot, the outstation responds immediately with a null response having no data objects, and it is the master's responsibility to poll for events (assuming unsolicited reporting is not used). The master can issue polls either for event classes or for the specific event object expected, depending on system configuration, until the expected response object is returned, or a local timer in the master times out.

NOTE

System designers need to coordinate which polls the master should issue while waiting for event file objects in the response. The outstation needs configuration as whether to return file event objects in polls for class event data. See Clause 4. The master should accommodate whichever way the outstation is configured.

It is permissible to use unsolicited reporting to return file event objects if the master and outstation devices support it and the system designer chooses this alternative.

To open a file, the master issues a message using function code OPEN_FILE. This message requires File Command object, g70v3, containing the appropriate information. See Clause 4 for details. A File Command Status event object, g70v4, is returned. If the status code in the returned object is zero, the file was successfully opened, and the application should use the returned 32-bit file handle during ***write*** requests to the file until it is closed. A non-zero status code indicates that the file was not opened and that the file handle value is invalid.

The maximum block size permitted for transferring file data is negotiated during the open process. For writing a file, the master states the maximum number of octets it can transfer in the File Command object, g70v3, based on its own limitations. The outstation shall set a size, which is the lesser of the value in the master request or a limitation in the outstation, into the *maximum block size* field in the response's File Command Status object, g70v4. The master shall not transfer more octets than the block size returned from the outstation.

To write data, the master issues one or more ***write*** requests using a File Transport object, g70v5, specifying the file handle, the block number, and the data octets to be written. The block number starts at 0 in the first ***write*** request and shall increment by 1 for each subsequent ***write*** request. The master shall set the *last* bit in the last block.

If the outstation is unable to save the data immediately, it shall return a null response (no data objects), and after the data is saved, it returns a File Transport Status event object, g70v6, with an appropriate status code to indicate the success of the operation. However, if the outstation is fast enough, it may return the File Transport Status immediately as the response to the ***write*** request.

NOTE

The outstation should not assume that each block, except the last, has an identical size. Both the outstation and the master should update their local file offset indicators after each ***write*** request.

If there is an error during the write process, the status code in the File Transport Status event object, g70v6, shall indicate the problem.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

After all of the file data has been transferred to the outstation, the master issues a ***close file*** request having a File Command Status object, g70v4. This object is not an event object when used in the request. The outstation shall respond with a message containing an identical File Command Status object as that which appeared in the request.

EX 38	This is an example of a file ***write*** request.

►►► Request Message (beginning)

C3	02	46	05	5B	01	13	00	66	77	88
AC	FC	Grp	Var	Qual	Range	Size prefix		←		

►►► Continuation of Request Message

99	02	00	00	80	A5	A5	A5	A5	A5	A5
				File Transport Status object						

►►► Continuation of Request Message

A5	A5	A5	A5	A5	
				→	

◄◄◄ Response Message (beginning)

E3	81	00	00	46	06	5B	01	09	00	66
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Size prefix		←

◄◄◄ Continuation of Response Message

77	88	99	02	00	00	80	00	
		File Transport Status object					→	

►►► Confirm Message

C3	00
AC	FC

5.3.5 Retrieving individual file information

The master issues a request to the outstation using function code GET_FILE_INFO with a File Descriptor object, g70v7. The outstation responds with the requested information in a File Descriptor object. This is described in Clause 4.

If the file is a directory, the *file type* field in the response shall be set to 0 and the *file size* field specifies the number of entries in that sub-directory excluding any links to itself or its parent.

5.3.6 Retrieving file directory information

Directory files are special types of files that hold information about all of the files within organizational units called “directories” and “sub-directories.” Each directory file contains information about a set of 0 or more standard files and other directory files, which are known as sub-directories. Regardless of how a

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

device internally organizes its filing system, it shall have the ability to map to directory files. The term *empty directory* refers to directories that contain no information about other files or sub-directories.

Information can be retrieved for all the files in a directory by reading the directory file. The ***read*** request is as described in 5.3.2 where the file handle is that obtained during the ***open file*** request, which contained the name of the directory file. The response to the ***read*** request contains a File Transport object, g70v5, for each directory file referenced in the ***read*** request.

The data returned in File Transport objects for directory files is formatted similar to a sequence of File Descriptor objects, g70v7. An object header is not included in the data, and there is one "file descriptor element" for each file. Also, if one or more of the returned files are themselves sub-directories, their *file type* fields are 0 and their *file size* fields indicate the number of files in the respective sub-directory file. The file size reported for empty directories and sub-directories shall be 0.

The *file name string* fields in the returned file descriptor elements only contain the base file name and do not include full path information as that information is already known from the File Descriptor object in the ***read*** request.

EX 39	Assume that an outstation has the following directory structure:
	
	file0
	dir1
	  file1
	  file2
	  dir2
	  file3
	dir3
	  file4
	
	NOTE—nn in the tables that follow represent appropriate hexadecimal numbers whose exact values are not important for the sake of illustration.

If the master were to read the root directory file, it would receive data in the File Transport object that appears like this (least significant octets are at left side of field):

Filename offset (hex)	Filename size (hex)	File type (hex)	File size (hex)	Time of creation (hex)	Permissions (hex)	Request Id (hex)	File name octets (ASCII)
14 00	05 00	01 00	nn nn nn nn	nn nn nn nn nn nn	00 00	nn nn	file0
14 00	04 00	00 00	03 00 00 00	nn nn nn nn nn nn	00 00	nn nn	dir1
14 00	04 00	00 00	01 00 00 00	nn nn nn nn nn nn	00 00	nn nn	dir3

If the master were to read the dir1 file, it would receive data in the File Transport object that appears like this (least significant octets are at left side of field):

Filename offset (hex)	Filename size (hex)	File type (hex)	File size (hex)	Time of creation (hex)	Permissions (hex)	Request Id (hex)	File name octets (ASCII)
14 00	05 00	01 00	nn nn nn nn	nn nn nn nn nn nn	00 00	nn nn	file1
14 00	05 00	01 00	nn nn nn nn	nn nn nn nn nn nn	00 00	nn nn	file2
14 00	04 00	00 00	01 00 00 00	nn nn nn nn nn nn	00 00	nn nn	dir2

When an empty directory file is read, the response contains a File Transport object header as usual, but the associated File Transport object has no data octets.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5.3.7 Deleting a file

This clause provides an overview of the procedures for deleting a file. The reader should refer to the function code descriptions in 4.4 for details of each operation. Deleting a file consists of the following transactions:

— Optional authentication.

— Delete file.

If the system requires an authentication key to delete a file, the master issues an ***authentication*** request as outlined in 5.3.1.

The master issues a request to the outstation using function code DELETE_FILE with a File Command object, g70v3. The outstation responds when a File Command Status event object, g70v4, is returned. This is described in Clause 4.

5.3.8 Rules relating to files

— **Rule 1:** As is the case for all event objects returned from the outstation, the outstation shall request an Application Layer confirmation from the master.

— **Rule 2:** If a file is opened for non-appended writing, it shall be truncated to zero length.

— **Rule 3:** If a file is opened for writing, the time of creation and permission fields in the File Command object, g70v3, that appears in the **open file** request shall be used as attributes for the file in the outstation.

— **Rule 4:** The outstation may reject requests to create files with permissions that it does not allow or cannot support.

— **Rule 5:** Outstations are not required to support appended writing.

— **Rule 6:** If the end of file is indicated in a device receiving a file before the expected file size is satisfied, the receiving device shall treat the file as completely received and not consider this to be an error condition.

— **Rule 7:** If the end of a file is longer (has more octets) than the file size expected, the receiving device may treat this as an error and reject the transfer.

— **Rule 8:** A file size of 0xFFFFFFFF is permitted in a File Command object in a **write** request and indicates the actual size is unknown. Devices are not required to accept an unknown file size.

— **Rule 9:** File handles are unique to an outstation. No assumptions should be made regarding handle numbers, and there shall be no correlation between the handle number and any function or specific set of data. Outstations shall not reuse file handles except after a restart.

— **Rule 10:** Devices may limit the number of files simultaneously opened to any number including just one.

— **Rule 11:** An opened file may not be deleted. If the master attempts to delete an open file, the outstation shall return a file locked error.

— **Rule 12:** If a file is opened and there is no activity referencing the file handle, the outstation shall automatically close the file. The timeout value shall be configurable up to one hour. The final state of a file closed due to activity timeout is undefined. When this condition occurs, the outstation shall

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

send a File Transport Status object using a status code value of HANDLE_EXPIRED (see Data Object Library).

— **Rule 13:** When an error is detected by an outstation upon receipt of a **read** request, the outstation shall automatically close the file and force the file handle invalid.

— **Rule 14:** When an error is detected by an outstation upon receipt of a **write** request, the outstation shall automatically close the file and force the file handle invalid. The contents of the file in the outstation are indeterminate.

— **Rule 15:** Directory files may only be opened for reading. Masters are not permitted to write them.

— **Rule 16:** The root directory may not be deleted. The vendor may choose whether none, some, or all sub-directories may be deleted via DNP3. If a vendor allows the master to delete a directory, the vendor may also specify whether the included files and sub-directories shall be deleted first, or whether deleting the directory also deletes all of its files and sub-directories.

5.4 Data sets

5.4.1 Preliminary background

It is advantageous in some systems for DNP3 devices to be able to transport structured data that consists of a possibly non-homogeneous collection of data types. Prior to data sets, the emphasis was on transporting simple binary inputs, analog and counter values moving toward the master and control and analog outputs moving toward the field.

For example, an electrical system relay might have something called a virtual fault object that it uses to report the system conditions when a fault event occurs. This object might contain the current in all three phases immediately before a breaker operation, which parameters triggered the breaker operation, the fault classification, and the computed distance to the fault.

One advantage of data sets is that the data can be kept together so that snapshots of an object are available instead of attempting to form a view of an object from different pieces of data that were collected at different times and the necessity of having to put the pieces together at the receiving end.

The values placed into a data set are not always available for reading on demand. In the fault example, the values represent a transient situation, one that only occurs upon detection of a fault. There is no way to record the distance to a fault until there is a fault. There are also situations where the data does not fit neatly into one of the existing point types such as analog inputs, binary inputs, or counters; the type of gas transported in a pipeline is an example.

It is helpful to see the contents of a few sample data sets prior to getting into formal details. Using the electrical system fault from above as an example, the collection[19] of data might contain the elements shown in Table 19.

[19] The term "collection" is used in this document as a noun to represent a group or set of data. It was chosen to avoid confusion with other terms that have a specific meaning in DNP3.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 19—Electrical fault data set

Data type	Description
Time	When fault occurred.
Floating-point	Current phase A.
Floating-point	Current phase B.
Floating-point	Current phase C.
Floating-point	Ground current.
Bit string	Parameters that triggered breaker.
Text string	Fault classification.
Integer	One-hundredths of kilometer distance from substation.

There are eight data elements. Each has a basic data type—time, floating-point value, bit string, text string, and integer.

An electrically operated pump is another example of a data set (Table 20).

Table 20—Pump-valve data set example

Data type	Description
Integer	Open position 0 to 100 percent.
Floating-point	Flow rate.
Integer	Downstream pressure.
Integer	Fluid temperature.
Floating-point	Fluid's specific gravity.
Integer	Motor voltage.
Integer	Motor current.
Integer	RPM.
Integer (boolean)	Operating (on/off state).

There are many real-world examples of where the same data set structure is required repetitively. Consider an electrical feeder. If a data set structure were defined to include the three-phase voltages, currents, watts, vars, etc., it could be reused for each of the multiple feeders in a substation. Another example is a data set structure for conveying the first 32 harmonics on an electrical phase wire. The same structure is usable for each of the three phases. Lastly, consider the water wells in many municipalities. If there were a data set structure defined for a well, the same structure applies to each well around the city.

5.4.2 Data set, data set descriptor, and data set prototype overview

DNP3 uses three basic constructs to describe and convey data sets: a *Data Set*, a *Data Set Descriptor*, and a *Data Set Prototype*.

5.4.2.1 Data set

A *Data Set* is a construct that conveys the actual values in the collection. Data sets also contain an identifier to indicate which specific data collection and at least one time value, the time when the data set was created. A data set is used during runtime for transferring a collection of data values from the outstation to the master, or vice versa. To minimize the communications bandwidth, a data set only contains enough overhead octets for the receiver to parse the object.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Data sets are constructed from a sequence of elements. Each element provides a single value.

The specific DNP3 group numbers associated with data sets are 87 and 88.

5.4.2.2 Data set descriptor

A *Data Set Descriptor* is a construct containing meta-data describing the data appearing in a data set. Its purpose is to provide information regarding the structure, ordering, and type of data values within a data set.

Data set descriptors are constructed from a sequence of elements called descriptor elements. Each describes a data element in the data set, a reference to a data set prototype or a name attribute.

The specific DNP3 group number associated with data set descriptors is 86.

NOTE—From here on in 5.4, as a shorthand expression, when the term *descriptor* is used alone, it refers to *data set descriptor*.

5.4.2.3 Data set prototype

A *Data Set Prototype* is a construct that contains a list of descriptor elements that define a group or subset of elements that exist in a data set. Reference to a prototype appearing in a data set descriptor provides a shorthand method for indicating that the data set contains elements described by the list of descriptors in the prototype.

Prototypes serve two main purposes. They allow vendors and third-party standards groups to predefine a subset of elements that shall appear within a data set. Second, they reduce the number of octets in messages transporting data set descriptors when the same pattern would otherwise appear repeatedly.

The specific DNP3 group number associated with data set prototypes is 85.

NOTE—From here on in 5.4, as a shorthand expression, when the term *prototype* is used alone, it refers to *data set prototype*.

5.4.2.4 Relationship of data sets, data set descriptors, and data set prototypes

Data sets and descriptors occur in pairs. Data set prototypes are optional.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 22—Relationships

The descriptor provides information needed by the receiving device to interpret the sequence and type of elements in the data set.

A data set transfers values during normal runtime. Its element ordering is as specified by the corresponding data set descriptor.

A data set descriptor may reference one or more prototypes instead of including the descriptor elements from the prototype.

5.4.2.5 Read, write, and control

Depending on a data set's purpose or reason for existing, a master can

— Request to read a data set.

— Request to write a data set.

— Issue a control command incorporating a data set object. See 5.4.11 for an in-depth discussion about control commands.

A few examples: Data sets that convey values measured in the outstation are readable by the master. A master writes new parametric values to an outstation configuration data set and reads it back to confirm the values. A master issues a select–operate command to load new integral, derivative, and gain factors for a PID algorithm.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5.4.2.6 Data set and data set prototype definition

Data sets and data set prototypes are defined by an outstation or by a master depending on which is responsible for the structural design.

5.4.2.6.1 Outstation defined

A data set is defined by an outstation if the outstation determines its structure.

Before transferring a data set defined by an outstation, the master shall know how its data is organized. A data set descriptor contains this information, and the master shall read the data set descriptor from the outstation. If a data set descriptor references a data set prototype, the master shall read the prototype from the outstation.

NOTE—It is permissible for masters to save descriptor information in cache or configuration memory so that reading data set descriptors is not necessary.

Examples of data sets defined by the outstation are the faults and pump-valves discussed in 5.4.1 and outstation configuration. In these situations, the outstation defines the data structures.

The outstation may not dynamically define data sets after it restarts. Outstation-defined data sets are determined by configuration. If new configuration is downloaded from the master (via file transfer or other means) that adds new, or revises or removes existing, outstation-defined data sets, the outstation shall restart its DNP3 application (and set IIN1.7) prior to the outstation using the changed definitions.

The outstation may not dynamically define data set prototypes after it restarts. Outstation-defined prototypes are determined by configuration. If new configuration is downloaded from the master (via file transfer or other means) that adds new, or revises or removes existing, outstation-defined data set prototypes, the outstation shall restart its DNP3 application (and set IIN1.7) prior to the outstation using the changed definitions.

5.4.2.6.2 Master defined

A data set is defined by a master if the master determines its structure.

In some situations, a master can define a data set in the outstation. Before an outstation can parse a data set defined by the master, the outstation shall know how its data is organized. A data set descriptor contains this information, and the master shall write the data set descriptor to the outstation. If the data set descriptor references a data set prototype, then the master shall also write the prototype if it does not already exist in the outstation.

A data set containing a sequence of algorithmic values executable in a local automation application would be an example of a data set defined by master. In this example, the outstation cannot know ahead of time what steps it should execute until the master conveys them.

Data sets defined by the master shall be readable and writable. This allows the master to read back, for verification purposes, the same information that it wrote.

The outstation shall not change the definitions of master-defined data sets or data set prototypes.

Masters and outstations are not required to support data sets or prototypes defined by the master. This ability is optional, even for devices that support data sets and prototypes defined by the outstation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5.4.2.7 Transmission sequence

Data sets are transmitted when appropriate. An outstation may send an event data set when an event occurs, or it shall send a static (present value) data set if requested by the master. A master may write data sets when it has new information for the outstation.

Before a receiver can parse a data set, it is necessary for the receiver to know the data organization within the data set. Data set descriptors contain this information, and therefore, for data sets defined by an outstation, the master shall read the data set descriptors from the outstation, or retrieve them from cache or configuration memory. Similarly, for data sets defined by the master, the master shall write the data set descriptors to the outstation before that master can transmit the data sets.

If a data set descriptor contains a reference to a data set prototype, then it is necessary to exchange the prototype. If both receiving and transmitting devices have knowledge of the prototypes, from cache, configuration tables, or reading them from a web server, it is not necessary to exchange the prototypes at startup.

It is not mandatory to request a read or a write of descriptors immediately following an outstation restart; however, it is advisable to do so if the outstation transmits event data sets. It is not required that data descriptors be written or read if they are guaranteed to be known by both ends after the master or outstation restarts.

After a restart, outstations are not required to remember data set descriptors written to them previously by a master. Likewise, masters are not required to remember descriptors they uploaded before a restart. Masters and outstations shall be able to transmit their descriptors even if it is unnecessary in certain systems. Devices may not assume that the other end shall never need transference of data set descriptors. This provides greater assurance of interoperability.

Because the master station can restart at any time without the outstation being aware, outstations shall be prepared to receive requests to read or write descriptors at any time, even though similar requests were received previously.

Data set prototypes are more likely to be standardized and pre-defined than are data set descriptors, and there is reason to believe some shall be available in libraries. It is not necessary to exchange prototypes if they are guaranteed to be known by both ends after the master or outstation restarts. As with descriptors, masters and outstations shall be able to transmit their prototypes even if it is unnecessary in certain systems. Devices may not assume that the other end shall never need transference of data set prototypes. This provides greater assurance of interoperability.

Masters may write descriptors and prototypes in any order. Outstations shall not assume that master stations shall write data set prototypes before writing data set descriptors to the outstation.

Outstations shall reject a master request to write a data set if the outstation does not have knowledge of the complete data set structure, which is obtained from the data set descriptor and any data set prototypes the descriptor references. It does this by setting IIN2.2, PARAMETER_ERROR in the response.

Outstations shall accept a master-defined prototype that duplicates an already existing outstation-defined prototype, the only difference being the ID. This includes the ability for the master to read back the data set prototype with the same ID that it previously wrote.

Because the master station can restart at any time without the outstation being aware, outstations shall be prepared to receive requests to read or write prototypes at any time, even though similar requests were previously received.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5.4.2.8 Identifiers

5.4.2.8.1 Data set and data set descriptor identifiers

The identifier in a data set is the key that a parser uses to interpret the data set contents according to the information in the corresponding descriptor. Data set identifiers have integer values and differentiate one data set from another. They also serve as index numbers when a master requests to read a specific data set descriptor or data set.

The same identifier shall be used for a data set that is readable and writable. If the master does not already know which descriptors are in the outstation, it shall first read all of the descriptors for the data sets defined by the outstation, and then for data sets defined by the master, transmit the master's unique descriptors.

The master shall use the identifiers appearing in descriptors obtained from the master's initial read of data set descriptors from the outstation. The master may not re-number or provide duplicate identifiers for those data sets that are both read and written.

Data set identifiers shall be contiguous. An outstation shall begin its numbering from zero and count upward without gaps. The burden of assuring that identifiers are sequential for the data sets defined by a master belongs to the master. When a master station defines data sets, the identifier numbers shall begin where the outstation left off. For example, the outstation defines N data sets and the master defines M data sets. The outstation sends descriptors 0 through (N – 1) when the master requests to read its descriptors. The master shall assign identifier N to its first descriptor, (N + 1) to its second, and so on, and writes them to the outstation.

Because data sets can be unique from outstation-to-outstation and from vendor-to-vendor, each data set definition has an integer identifier that is unique to an outstation–master pair. Masters that communicate with multiple outstations shall keep separate lists of identifiers for each outstation. Similarly, outstations that communicate with multiple masters shall keep separate lists of identifiers for each master.

It is not permitted to require a specific index ordering of data sets. An outstation is permitted to assign any index to any data set as long as the indexes are contiguous and numbering begins with zero.

To aid the master station in determining the number of data sets that are available in the outstation, the outstation shall support Device Attributes, group 0, variations 214 and 215, at index 0.

5.4.2.8.2 Data set prototype identifiers

The identifier illustrated in Figure 22 of a data set prototype is unrelated to the identifier of the data set descriptors that reference the prototype. Prototype identifiers have integer values and indicate a sequential index into a list of prototypes in a specific outstation,; they are used as index numbers when a master requests to read or write a specific prototype.

The same identifier shall be used for a data set prototype that is readable and writable. If the master does not already know which prototypes are in the outstation, it shall first read all of the prototypes defined by the outstation, and then for data set prototypes defined by the master, transmit the master's prototypes.

Prototype identifiers shall be contiguous. An outstation shall begin its numbering from zero and count upward without gaps. The burden of assuring that identifiers are sequential for the data sets defined by a master belongs to the master. When a master station defines data set prototypes, the identifier numbers shall begin where the outstation left off. For example, the outstation defines N data set prototypes and the master defines M prototypes. The outstation sends descriptors 0 through (N – 1) when the master requests

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

to read its prototypes. The master shall assign identifier N to its first prototype, (N + 1) to its second, and so on, and writes them to the outstation.

To aid the master station in determining the number of data set prototypes that are available in the outstation, the outstation shall support Device Attributes, group 0, variations 212 and 213, at index 0.

5.4.3 Data sets are application specific

DNP3 does not attempt to define a suite of standard data sets. Data set definition depends on the specific application, and there are far too many possibilities for DNP3 to attempt standardizing them all.

This does not preclude special interest groups from developing pre-defined data sets and data set prototypes. For example, it may be possible for a relaying group to define a standard fault data set prototype. Likewise, it might be achievable to define collections of data sets that permit DNP3 to transport data from non-DNP3, object-oriented protocols. The definition of those data sets is outside the scope of this standard.

5.4.4 Data set details

5.4.4.1 List of elements

Data sets consist of a list of data elements that contain data. The first element is always the data set identifier, the second element always contains the time of creation, and the other elements contain data values.

— A data set is conveyed in a DNP3 object.

— Each data set shall fit within a single Application Layer fragment.

— Within a single outstation, event and static data sets that relate to the same collection are based on the same data set descriptor and have an identical object format.

The first element, the identifier, provides the correspondence to a data set descriptor and serves as an index as previously discussed in 5.4.2.8.1.

The second element is the time of creation. This refers to the time associated with the data elements that follow. If the data set is an event data set, the time of creation is the time when the event occurred. If the data set is a static data set, it is the common time when all of the subsequent data element values were put together into the data set.

All of the other elements, beginning with the third, contain data values as specified from the corresponding data set descriptor.

5.4.4.2 Event and static data sets

Data sets are applicable to event and static conditions. Events relate to when *something of interest* occurs. DNP3 does not define what “something” is because “something” is application specific. In the example of an electrical fault, the “something” is the occurrence of a fault condition. In other situations, a system may require a snapshot of certain values stored at 15-minute intervals during a loss of communications; the end of a 15-minute interval is the “something” of interest.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Static data relates to present conditions. The master may poll for specific data sets in order to update operator displays or for other reasons. The master may write a data set to establish configuration parameters or for another reason.

Not all data sets have a static value because a complete data set may not exist until an event occurs. For example, the elements of an electrical fault data set only have meaningful values at the instant when a fault occurs.

Data sets defined by the master are always static. An outstation shall not generate events for this type.

5.4.4.3 Data set names

Each data set definition may have a text string name. Data set names do not appear in the data set itself but may optionally appear in their corresponding data set descriptors.

Subclause 5.4.8 provides more information regarding naming.

5.4.4.4 Data set identifiers

Within a single outstation, event and static data sets that relate to the same collection, use the same identifier.

5.4.4.5 Data type codes

Data sets are limited to the data types in Table 21 for the values that appear in each of its elements.

Table 21—Data type codes

Code	Code name	Description
	NONE	Data type does not exist or a placeholder.
	√ VSTR	Visible ASCII characters suitable for print and display.
	√ UINT	Unsigned integer.
	√ INT	Signed integer.
	√ FLT	Floating-point.
	√ OSTR	Octet string.
	√ BSTR	Bit string.
	√ TIME	DNP3 time.
	UNCD	Unicode strings (requires two octets per character).

Regarding data types:

— Data type codes do not specify the number of octets required to convey the value. That information is encoded in the message by a length octet.

— Floating-point types are limited to 4 and 8 octets for transporting 32-bit and 64-bit floating-point values. No other sizes are acceptable. Implementers that use 64-bit floating-point values shall carefully consider the consequences of possible non-interoperability.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— The preferred size for integer types is 1, 2, and 4 octets (8, 16, and 32 bits). Implementers that use integers having more than four octets shall carefully consider the consequences of possible non-interoperability.

— The bit alignment in bit string types requires the first bit to appear in bit 0 of the first octet of the value. If the number of bits in a bit string is not an integral of 8, the unused bits in the last octet of the value shall be cleared to 0.

— Implementers that use Unicode shall carefully consider the consequences of possible non-interoperability in that this data type is not universally supported.

— Master devices and outstations that accept data sets defined by the master shall support

1) Those data type codes marked with a √ symbol in Table 21.
2) 32-bit integers (types UINT and INT of size 4 octets).
3) 32-bit floating-point values.

5.4.5 Descriptor elements

5.4.5.1 Definitions

Before discussing descriptor elements, it is best to define two terms, namespaces and Universally Unique Identifier (UUID).

5.4.5.1.1 Namespaces

A namespace is a reference to a set of names that are in some way related to each other. They are helpful to prevent confusion if a name such as “Distribution Feeder” appears in two or more named spaces.

Namespaces are used to qualify the names used for data set prototypes. Data set prototypes are most often given generic names such as “Fault,” “Well Pump,” and “Generator.” Using a namespace reduces the confusion if two organizations want the same name for their prototype. With a namespace, the World-Wide Relaying Organization and the Wind Power Association (both fictitious groups) can define a DNP3 data set prototype with the name of “Distribution Feeder” without concern for user misinterpretation.

Namespaces are intended to reflect the name of a standards organization, a standards specification, such as “IEC 99099-9,” or a vendor, but that is not a requirement.

Namespaces shall be registered on the DNP3 Web site[20] in order to assure unique namespaces. A registered namespace text string is public information. It is preferred that names within the namespace be made public by the registering organization for the sake of interoperability, but they may be kept as private information.

There are several rules for generating acceptable namespaces:

— The namespace shall consist of ASCII letters, digits, spaces, underscores, and hyphens [`A..Z,a..z,0..9, ,_,-`]. No other characters are permitted.

— Namespaces shall begin with a letter or digit.

[20] http://www.dnp.org

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— Namespaces are case sensitive; however, when checking for uniqueness, a case-insensitive comparison shall be used. For example, "`CaT`" is unacceptable if "`Cat`" were already reserved.

— The namespace may not contain two or more contiguous space characters, two or more contiguous underscores, or two or more contiguous dashes. For example, "`My--Name`" is not acceptable.

— The number of characters in the namespace shall not exceed 32.

— Names shall not use profanity or vulgar terms. This includes those that would be considered racist, hateful, sexual, or obscene in nature.

5.4.5.1.2 UUID

A UUID is a unique 16-octet string.

There is a program available on the DNP3 Web site (see footnote 21) that generates UUIDs. This is the only approved method for generating UUIDs for DNP3; no other generators are acceptable. The reason is because several methods exist for generating UUIDs, and to provide uniqueness, only one is permitted.

For human readability, UUIDs are often expressed as an ASCII string that includes the hexadecimal characters, hyphens, optional opening and closing curly brackets, and spaces, as in

```
{ C2F41010-65B3-11D1-A29F-00AA-00C15A82 }
```

Only 16 octets are transmitted. Each pair of hexadecimal characters in the human-readable format represents the contents of one octet in the 16-octet string.[21] The leftmost characters, "C2" in the example, are transmitted first in an octet having a hexadecimal value of 0xC2 or 194 decimal. The rightmost characters, "82" in the example, are transmitted in the 16th octet; its value is 0x82 or 130 decimal.

5.4.5.2 Descriptor element overview

Data set descriptors and data set prototypes are formed by a sequence of descriptor elements. Each descriptor element contains information about

— A data element in the corresponding data set.

— A reference to a data set prototype.

— A namespace or name attribute of a data set or a data set prototype.

Subclauses 5.4.6 and 5.4.7 describe how descriptor elements are used.

Specifically, each descriptor contains

— The length of the descriptor element.

— The type of descriptor element.

— A data type code.

[21] Computer programs written in C define a UUID as a structure consisting of a 32-bit unsigned long, two 16-bit unsigned short integer and an array of 8 characters. In an effort to define a UUID in DNP3 that is independent of the byte ordering used by a CPU, this specification requires transmitting octets in the same order as the ASCII string representation of a UUID. For the example in 5.4.5.1.2, the "C2F41010" represents the hexadecimal value, 0xC2F41010, of the 32-bit unsigned long value in the structure.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— The maximum data length.

— An optional ancillary value.

These are discussed as follows.

5.4.5.2.1 Descriptor element length

This is a single octet that a parser uses to learn the length, in octets, of the entire descriptor element.

5.4.5.2.2 Descriptor element type

There are different types of descriptor elements; these are enumerated by the code numbers in Table 22.

Table 22—Descriptor codes

Code	Code name	Description
1	ID	Specifies an identifier attribute associated with a descriptor. Each descriptor has a unique identifier to differentiate it from all others. The ancillary value associated with this field is a UINT. The value is the identifier integer.
2	UUID	Specifies a UUID associated with a data set prototype. The ancillary value associated with this field is an OSTR of exactly 16 octets that contain a UUID.
3	NSPC	Specifies a namespace attribute associated with a data set prototype. The ancillary value associated with this field is a VSTR. It is the namespace.
4	NAME	Specifies a name attribute associated with the data set or data set prototype. The ancillary value associated with this field is a VSTR. It is the data set's or data prototype's name.
5	DAEL	Specifies the occurrence in the data set of a simple data element. The ancillary value associated with this field is a VSTR. It is optional and specifies the name associated with the data value in the data set.
6	PTYP	Specifies the occurrence in the data set of a subset of elements defined in a data set prototype. The ancillary value associated with this field is an OSTR that is the concatenation of a 16-octet UUID followed by an optional prototype instance name VSTR.
7	CTLV	Specifies the occurrence in the data set of an element containing a value to be issued to the receiver when the data set appears in SELECT, OPERATE, DIRECT_OPERATE, and DIRECT_OPERATE_NR commands. The ancillary value associated with this field is a VSTR. It is optional and specifies the name associated with the value in the data set.
8	CTLS	Specifies the occurrence in the data set of an element intended to convey whether the subsequent CTLV values should be issued in SELECT, OPERATE, DIRECT_OPERATE, or DIRECT_OPERATE_NR commands, and to convey the control status in the corresponding response messages. The ancillary value associated with this field is a VSTR. It is optional and specifies the name associated with the data value in the data set.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5.4.5.2.3 Data type code

This is a single octet that specifies the general format for which the data in the corresponding element of the data set shall be transmitted. The data type codes are specified in Table 21.

5.4.5.2.4 Maximum data length

This is a single octet specifying the maximum number of octets permitted to express the data or control value in the corresponding data set. It provides information for the receiver to allocate memory or otherwise limit some aspect of the parsing activities.

5.4.5.2.5 Ancillary value

The ancillary value completes the description when the codes are not enough.

The ancillary values depend on the descriptor element type, as described in Table 22, and the data set, prototype, or data element to which they refer.

5.4.6 Data set descriptor details

5.4.6.1 General

Data set descriptors specify the sequence of elements that form data sets.

— A data set descriptor is conveyed in a DNP3 object.

— Each data set descriptor shall fit within a single Application Layer fragment.

5.4.6.2 Data set descriptor construction

Descriptors are constructed from a sequence of descriptor elements. Descriptor elements are used to specify several things—the data set descriptor identification, the elements in a corresponding data set, a subset of data elements as found in a data set prototype, and a name. Descriptor elements are discussed in 5.4.5.

5.4.6.2.1 First descriptor element

The first descriptor element in a data set descriptor is always an ID element.

5.4.6.2.2 Name descriptor element

If the data set has a unique name, that name shall appear as the second descriptor element. The name is specific to the data set having the same identifier as is specified by the ID element. It is specified using a NAME type.

To illustrate, assume there is an RTU at all 200 pumping stations along a gas pipeline, and there is a data set defined for each. All 200 data sets could utilize the identical construction in that they all have the same sequence of elements that specify pressure, temperature, flow rate, etc. It might be advantageous for the

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

system engineer to name each data set—"Pump Station 1," "Pump Station 2," … or "Fox Hill," "Lookout Point," … Those names might best describe the data set.

On a smaller geographic scale, each feeder of an 8-feeder substation may have a different data set name: "Feeder 101," "Feeder 102," … or "Route 661," "County Line Road," …

Subclause 5.4.8 provides more information regarding naming.

It is not required to have names for data sets and inclusion of a NAME descriptor element is optional.

5.4.6.2.3 Data descriptor elements

A data set descriptor may include data element descriptors of type DAEL. Each DAEL element that appears in a data set descriptor specifies a single data value in the corresponding data set.

It is not necessary to have any DAEL elements if the data set shall only be used for control operations, although it is permissible to include them.

5.4.6.2.4 Control-related descriptor elements

A data set descriptor may include control-related element descriptors of type CTLV and CTLS. These are used to specify elements to be controlled when issued in commands using SELECT, OPERATE, DIRECT_OPERATE, and DIRECT_OPERATE_NR function codes. Each CTLV and CTLS element that appears in a data set descriptor specifies a single value in the corresponding data set.

5.4.6.2.5 Prototype reference descriptor elements

A PTYP element may be included in a data set descriptor as an alias for a sequence of DAEL, CTLV, or CTLS descriptor elements. A PTYP element references a data set prototype.

When a PTYP appears, software shall use the DAEL, CTLV, and CTLS descriptor elements from the prototype to determine the data type and ordering of the elements in the corresponding data set. In essence, the PTYP implies that the elements from the prototype are to be substituted for the PTYP element.

It is acceptable for the same PTYP to appear multiple times in the same descriptor. For example, if PTYP "xyz" defines the harmonics of a single phase, a data set descriptor designed for three-phase power line could have three "xyz" PTYP elements.

5.4.6.2.6 Prohibited descriptor elements

A data set descriptor may not include NSPC or UUID descriptor elements.

5.4.6.2.7 Data value ordering

It is permissible to intermix DAEL, CTLV, CTLS, and PTYP elements within in a descriptor. The order of data values in the data set is the same order that the DAEL, CTLV, and CTLS elements appear in the data set descriptor and in the included PTYP prototype elements.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5.4.7 Data set prototypes

5.4.7.1 General

Data set prototypes specify the sequence and type of a subset of data elements within a data set.

— A data set prototype is conveyed in a DNP3 object.

— Each data set prototype shall fit within a single Application Layer fragment.

5.4.7.2 Data set prototype construction

Prototypes are constructed from a sequence of descriptor elements. Descriptor elements are used to specify several things—the data set prototype identification, the UUID of the prototype, namespace and name of the prototype, and the data elements. Descriptor elements are discussed in 5.4.5.

5.4.7.2.1 First element

The first descriptor element in a data set prototype is always an ID element. It is not related to the ID member of a data set descriptor. It is used to access individual prototypes using classic DNP3 index numbers.

As an example, an outstation has a repertoire of 12 prototypes. The master could, if it so desired, request the fifth and sixth prototypes using indexes 4 and 5 in the range field of an Application Layer header with qualifier codes 0x00.

The first element, the ID element, is the only element in a prototype that is not considered when assigning the prototype's UUID. See 5.4.7.2.2.

5.4.7.2.2 Second element

The second descriptor element in a data set prototype is always a UUID element. It is the UUID associated with one and only one prototype. If any member of an existing prototype, except the ID element, should need to change in any respect, the new prototype shall have a new UUID. This includes namespaces, names, types, maximum lengths, and so forth.

When a standards organization or vendor defines a prototype, it shall also specify the UUID. Thus, given the same UUID, it shall always signify the exact same information regardless of where in the world it appears.

5.4.7.2.3 Third and fourth elements

If prototype namespace and name are used, they shall appear in the third and fourth descriptor elements; the third element shall be a NSPC element, and the fourth shall be a NAME. These refer to the namespace and name of the prototype, not the individual or specific data set with which it is associated.

Either NSPC and NAME appear, or they do not. It is not permissible to have a NSPC without a NAME or a NAME without a NSPC.

Subclause 5.4.8 provides more information regarding naming.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5.4.7.2.4 Data descriptor elements

A data set prototype shall include at least one data descriptor element of type DAEL if the prototype does not include a CTLV or CTLS element. Each DAEL element that appears in a data set prototype specifies a single data value in the corresponding data set.

5.4.7.2.5 Control-related descriptor elements

A data set descriptor may include control-related element descriptors of type CTLV and CTLS. These are used to specify elements to be controlled when issued in commands using SELECT, OPERATE, DIRECT_OPERATE, and DIRECT_OPERATE_NR function codes. Each CTLV and CTLS element that appears in a data set prototype specifies a single value in the corresponding data set.

5.4.7.2.6 Prohibited descriptor elements

A data set prototype may not have PTYP elements[22]; that is, prototypes may not be nested.

5.4.7.2.7 Data value ordering

The order of data values in the data set is the same order that the DAEL, CTLV, and CTLS elements appear in the data set prototype.

5.4.8 Naming guidelines

Names of various items are permitted within data set descriptors and data set prototypes. Names are helpful for human understanding and can serve as machine-readable identifiers when the names are carefully chosen.

Names are optional and are not required, but implementers should consider the following issues when choosing to provide names.

For the examples appearing in the following dashed items, assume that the implementer wants to create a data set descriptor for reporting the harmonics of a three-phase feeder. In doing so, he chooses to use a data set prototype that describes the magnitudes of the fundamental through the nth harmonic frequencies.

— The name of a prototype (see 5.4.7.2.3) should uniquely identify a single prototype within the same namespace. The name should be general so that it is applicable regardless of which specific data set descriptors reference the prototype or how many times the prototype is referenced within the same data set descriptor. For example, use the term “Harmonic magnitude” instead of “Phase A Harmonics.”

— The names associated with data elements within a prototype should be generic and not reference a specific instance of the prototype. For example, use “5th harmonic” and not “5th voltage harmonic” because the same prototype can then be used for describing voltage and current harmonics.

— The names associated with data set descriptors (see 5.4.6.2.2) should be specific to the data set. It is strongly suggested that the name be unique among all data sets within an outstation so that the name can be used as a synonym for the identifier element in the descriptor.

[22] This requirement may be relaxed in the future if the need arises. Until then, the prohibition against nesting means parsers do not need to incorporate recursion, which should minimize their complexity.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Avoid using the words "Data Set" in the name because these are redundant and minimizing the number of characters in a name is beneficial.

Continuing with the example, a suitable name for the data set would be "Harmonics feeder 105." This name includes an indication of which feeder the harmonics are associated and the word *Harmonics* narrowly defines the specific data set application.

— PTYP elements may contain a name in the ancillary value immediately following the UUID. This name should be specific and apply to the particular instance of a prototype. Because data set descriptors may include multiple prototypes, the name is useful for clarifying the differences. The example data set descriptor from above has three prototypes, one for each phase. Suitable names would be "Phase A," "Phase B," and "Phase C," or the names could be even more specific, "Phase A voltage," "Phase B voltage," and "Phase C voltage."

NOTE—The ancillary value actually contains the 16 UUID octets and the name characters.

When implementers use these guidelines for creating names, each element within a data set becomes identifiable by hierarchal name fields. As an example, the 3rd harmonic on phase B at feeder 105 can be referenced as "Harmonics feeder 105**.**Phase B**.**3rd harmonic" where a period is used to separate the fields.

In certain cases, implementers may choose to use a text UUID in lieu of the name. This permits a cryptic method of uniquely identifying a specific data value. Discussion of this technique is outside the scope of this document.

5.4.9 DNP3 object groups, classes, and indexes

5.4.9.1 Group numbers and class responses

Data sets, prototypes, and descriptors are transported using the group numbers specified in Table 23. The table indicates when the objects may appear in responses to requests for class data from the master.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 23—Data set related group numbers and report classes

Group	Description	Report classes
85	Data Set Prototype	May not appear in any class response.
86	Data Set Descriptor	May not appear in any class response.
87	Static Data Set	May appear in class 0 responses only if so configured.
88	Event Data Set	May appear in class 1, 2, or 3 responses.

5.4.9.2 Point indexes

The identifiers associated with data sets, descriptors, and prototypes correspond to index numbers—see 5.4.2.8 and the ID element type in 5.4.5.2.2.

Data set, descriptor and prototype transfers use qualifier code 0x5B in the object headers due to the variable size of those objects. This qualifier requires the range field to specify the number of objects, and that an octet count precede each object. There is no provision with this qualifier for a point index.

When the master requests to read a specific data set, descriptor, or prototype, it shall use a qualifier code that specifies a point index. In these situations, the object header's qualifier and field codes specify index numbers having the same values as the identifiers of the desired data sets, descriptors, or prototypes. Qualifier 0x00 is an example that specifies the objects are not prefixed, and the range field contains a start and stop index. A read request for group 86, variation 1, qualifier 0x00 and range field values 0x02 and 0x02 shall return the single data set descriptor with an identifier of 2.

5.4.9.3 Event generation and class assignment

If an outstation supports data set events,

— It shall be configurable to not send them. This is necessary for compatibility with masters that do not support data sets.

— It shall support function code 22, [ASSIGN_CLASS].

The master uses group 86 in the object header of an assign class request. (It does not use the static data type of group 87.)

5.4.10 Point index attributes

There are situations when it is desirable to correlate the elements of a data set to the indexes for the more primitive point types. For example, an IED that reports the power parameters on a 3-phase feeder data set can also report 9 individual analog input points in traditional class scans (3 voltage, 3 current, ground current, total megawatts and total megavars).

DNP3 provides a means to transport the correlation using a group 86, variation 3 object. This is a special variation of a data set descriptor. Each element in variation 3 contains a point type, and point index. The structure of group 86, variation 3 object is provided in the *DNP3 Data Object Library*. Point types are indicated by the group numbers shown in Table 24. If a value in the data set does not correlate to a point type in the classic database, group number 0 is used.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 24—Group numbers used for point types

Group (dec)	Point type
0	No point type is associated.
1	Binary input
3	Double-bit input
10	Control output
20	Counter
30	Analog input
41	Analog output
90	Application
101	BCD point
102	8-bit unsigned integer
110	Octet string
112	Virtual terminal

The elements in variation 3 appear in the same order as the DAEL descriptor elements of the data set descriptor.

The point indexes are intended as references only. In most situations, implementers should not use the values in a data set to update a master's point database. The reason is because the time relationship of the data set values is not synchronized with the timing of the normal database update mechanism.

Using analog inputs as an example, the master polls for static and event data and receives static analog input values in group 30 objects and events in group 32 objects. DNP3 does not specify when data sets are constructed, when the analog input values are obtained, or whether data sets shall be reported before or after analog input data. If the values from a data set were used to populate the same database values as the normal polling, it is possible that an older value would replace a more recently obtained value, and thus, the database would be inaccurate.

5.4.11 Control commands and responses

Data set objects of type g87v1 may appear in control requests and their responses. In general, controls refer to function codes SELECT, OPERATE, DIRECT_OPERATE, and DIRECT_OPERATE_NR and their responses.

Data sets used with control commands may contain multiple control values, that is, values that the outstation issues to the logical or physical end device. In some situations, the designer may want to control some of the values in one control command and control other values in the next command. For example, in a data set designed for cameras, one value sets the zoom, and two other values set the pan and tilt parameters. An operator may want to control the zoom independently from the pan and tilt. In addition, when pan and tilt values are set, they shall be output atomically (meaning together at the same instant in time).

The same data set may also include values that are not involved in the control but are useful for reporting the values of other parameters. The aperture opening and shutter speed in the preceding camera example are not controllable but are included in the data set for reporting from the outstation.

DNP3 does not specify what values a designer may include in a data set. The developer shall decide whether it is better to have a single data set that includes zoom, pan, tilt, aperture, and speed values or

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

whether it is best to have two data sets, one with zoom, pan, and tilt for controls and another for reporting aperture opening and shutter speed. DNP3 provides the capabilities needed for all of these possibilities.

The next paragraphs describe elements and rules for data sets used for controls.

5.4.11.1 CTLV, CTLS, and DAEL elements in control requests and responses

Table 25 describes the use of CTLV, CTLS, and DAEL elements in control requests and responses.

Table 25—Element types in control requests and responses

Element type	Controllable	Description
CTLV	Yes	Elements of this type represent the commanded values. In a request message, the values in the corresponding elements of a data set are the values that shall be issued to the receiving device, but only if the preceding CTLS element contains a value of zero. The values in response to SELECT, OPERATE, and DIRECT_OPERATE commands always mimic the values in the request. The master ignores the value in elements of this type in READ and UNSOLICITED responses, and the outstation ignores the value in elements of this type in WRITE requests.
CTLS	No	In request messages, elements of this type represent an indication to the receiver whether or not to issue the values in subsequent CTLV elements to the receiving device. A value of 0 indicates that the value shall be issued; a value of NON_PARTICIPATING (126) indicates that the receiving device shall not issue the values in subsequent CTLV elements to the receiving device. In responses to SELECT, OPERATE, and DIRECT_OPERATE commands, elements of this type represent a status code. The outstation always mimics the value in the request unless there is a reason why the command shall not succeed. In that situation, the value is set to an appropriate control status code as listed in part 1 of the Data Object Library Specification. The master ignores the value in elements of this type in READ and UNSOLICITED responses, and the outstation ignores the value in elements of this type in WRITE requests.
DAEL	No	Elements of this type may appear in data sets that also have CTLV and CTLS elements. However, DAEL elements are not intended to convey information in control requests and responses. The master and outstations ignore the value in elements of this type in SELECT, OPERATE, and DIRECT_OPERATE requests and responses. The outstation shall mimic whatever values the master places in the DAEL elements of the request when it sends the response, but otherwise it shall disregard its value.

5.4.11.2 Control status element (CTLS)

It is mandatory to include at least one control status element in any g87v1 object that is transmitted in a control command or response. A data set may include more than one CTLS if not all of the CTLV values are outputted atomically. A CTLS applies to all of the CTLV elements that follow it until either the next

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

CTLS element appears or the end of the data set occurs. The term applied to a CTLS and its subordinate CTLV elements is a *control-group*. Figure 23 shows a sample data set having two control-groups.

Figure 23 — Sample data set with CTLS and CTLV elements

Table 26 specifies the definition of a CTLS descriptor element.[23]

Table 26 —CTLS element structure and contents

Field contents	Field description
Depends on ancillary value	Descriptor element length
0x08	ID descriptor code
0x02	UINT
0x01	Maximum data length
Developer's choice	Ancillary value is name string

5.4.11.3 Control rules

The following rules apply when data sets are used in control commands having function codes SELECT, OPERATE, DIRECT_OPERATE, and DIRECT_OPERATE_NR and their responses

a) All of the standard select-before-operate rules **shall** apply. These include, but are not limited to, requirements for the ***operate*** request to exactly match the ***select*** request, and for the ***select*** and ***operate*** requests to have sequential sequence numbers in the application control octets. The complete set of rules is not repeated here; the reader is referred to the appropriate sections of the Application Layer specification.

b) The requirements stated in Table 25 apply.

c) Each control-group within a data set shall begin with a control status CTLS element. The structure and contents shall be as specified in Table 26. Developers may choose the name in the ancillary value field. Alternates or custom-designed CTLS elements are not permitted.

d) A data set may include multiple control groups.

[23] See objects g85v1 and g86v1 in the Data Object Library for a generic definition of descriptor elements.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

e) All of the CTLV values associated with a control group shall be issued atomically.

f) For data sets with multiple control groups:

   1) The outstation may issue the control values in each control group independently of the other control groups.

   2) The master prevents issuing control values for some of the control groups by setting the value of their respective CTLS elements to NON_PARTICIPATING (126).

g) If there is an error or a reason why the control operation shall not succeed for a CTLV element within a control group, the outstation reports the appropriate status code in only that control group's CTLS element. The outstation shall not alter CTLS elements due to errors detected for CTLV elements that are not members of the same control group.

   If the CTLS value in a request is NON_PARTICIPATING, the master does not want the outstation to issue the CTLV values in that control group; therefore, no errors can result, and the outstation shall return the same CTLS value in the response.

5.4.11.4 Message exchange illustration

Figure 24 provides an overview of a control message exchange. It includes a few other messages to demonstrate when g87v1 and g88v1 objects are used. The diagram is not intended to be comprehensive and only illustrates one out of many possibilities.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 24 — Example control message exchange

5.4.12 Example data descriptors, prototypes, and data sets

Table 27 through Table 30 illustrate the structure and contents of data set descriptors, data set prototypes, and data sets.

EX 40	Electrical fault example 1. Table 27 shows the construction of a data set descriptor for the electrical fault data set from Table 19. All of the descriptor elements are included in the data set descriptor, and prototypes are not used.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 27—Data set descriptor for electrical fault

Descriptor element		Element contents		Element description	
1st		0x05		Descriptor element length	Mandatory DNP3 identifier element
		0x01		ID descriptor code	
		0x00		No data type code (placeholder code)	
		0x00		Maximum data length	
		0x0000		Ancillary value is data set identifier	
2nd		0x11		Descriptor element length	Optional data set name
		0x04		NAME descriptor code	
		0x00		No data type code (placeholder code)	
		0x00		Maximum data length	
		“Fdr 11-A Fault”		Ancillary value is the data set name	
3rd		0x10		Descriptor element length	Current phase A
		0x05		DAEL descriptor code	
		0x04		FLT data type code	
		0x08		Maximum data length	
		“Current Phs A”		Ancillary value is the data element name	
4th		0x10		Descriptor element length	Current phase B
		0x05		DAEL descriptor code	
		0x04		FLT data type code	
		0x08		Maximum data length	
		“Current Phs B”		Ancillary value is the data element name	
5th		0x10		Descriptor element length	Current phase C
		0x05		DAEL descriptor code	
		0x04		FLT data type code	
		0x08		Maximum data length	
		“Current Phs C”		Ancillary value is data element name	
6th		0x0E		Descriptor element length	Current ground
		0x05		DAEL descriptor code	
		0x04		FLT data type code	
		0x08		Maximum data length	
		“Current Gnd”		Ancillary value is data element name	
7th		0x0B		Descriptor element length	Parameters that triggered breaker
		0x05		DAEL descriptor code	
		0x06		BSTR data type code	
		0x02		Maximum data length	
		“Triggers”		Ancillary value is data element name	
8th		0x0E		Descriptor element length	Fault classification
		0x05		DAEL descriptor code	
		0x01		VSTR data type code	
		0x08		Maximum data length	
		“Fault Class”		Ancillary value is data element name	
9th		0x07		Descriptor element length	1/100th kilometers distance to fault
		0x05		DAEL descriptor code	
		0x02		UINT data type code	
		0x04		Maximum data length	
		“Dist”		Ancillary value is data element name	

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

EX 41	Electrical fault example 2. Table 28 shows the construction of a data set descriptor for the electrical fault data set from Table 19. Table 29 shows the data set prototype that is referenced in the data set descriptor.

Table 28—Data set descriptor for electrical fault with prototype

Descriptor element		Element contents		Element description	
1st		0x05		Descriptor element length	Mandatory DNP3 identifier element
		0x01		ID descriptor code	
		0x00		No data type code (placeholder code)	
		0x00		Maximum data length	
		0x0000		Ancillary value is data set identifier	
2nd		0x11		Descriptor element length	Optional data set name
		0x04		NAME descriptor code	
		0x00		No data type code (placeholder code)	
		0x00		Maximum data length	
		“Fdr 11-A Fault”		Ancillary value is the data set name	
3rd		0x1D		Descriptor element length	Reference to a data set prototype
		0x06		PTYP descriptor code	
		0x00		No data type code (placeholder code)	
		0x00		Maximum data length	
		0xC2 0xF4 0x10 0x10 0x65 0xB3 0x11 0xD1 0xA2 0x9F 0x00 0xAA 0x00 0xC1 0x5A 0x82 “Fault Data”		Ancillary value is the mandatory UUID, which is followed by an optional instance name	

Table 29—Data set prototype for electrical fault

Descriptor element		Element contents		Element description	
1st		0x05		Descriptor element length	Mandatory DNP3 identifier element
		0x01		ID descriptor code	
		0x00		No data type code (placeholder code)	
		0x00		Maximum data length	
		0x0099		Ancillary value is prototype identifier	
2rd		0x13		Descriptor element length	UUID assigned to prototype
		0x02		UUID descriptor code	
		0x00		No data type code (placeholder code)	
		0x00		Maximum data length	
		0xC2 0xF4 0x10 0x10 0x65 0xB3 0x11 0xD1 0xA2 0x9F 0x00 0xAA 0x00 0xC1 0x5A 0x82		Ancillary value is the UUID	
3rd		0x15		Descriptor element length	Optional prototype namespace
		0x03		NSPC descriptor code	
		0x00		No data type code (placeholder code)	
		0x00		Maximum data length	
		“Earth Relaying Org”		Ancillary value is the data set name	

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Descriptor element		Element contents		Element description	
4th		0x0F		Descriptor element length	Optional prototype name
		0x04		NAME descriptor code	
		0x00		No data type code (placeholder code)	
		0x00		Maximum data length	
		"Feeder Fault"		Ancillary value is the data set name	
5th		0x10		Descriptor element length	Current phase A
		0x05		DAEL descriptor code	
		0x04		FLT data type code	
		0x08		Maximum data length	
		"Current Phs A"		Ancillary value is the data name	
6th		0x10		Descriptor element length	Current phase B
		0x05		DAEL descriptor code	
		0x04		FLT data type code	
		0x08		Maximum data length	
		"Current Phs B"		Ancillary value is data element name	
7th		0x10		Descriptor element length	Current phase C
		0x05		DAEL descriptor code	
		0x04		FLT data type code	
		0x08		Maximum data length	
		"Current Phs C"		Ancillary value is data element name	
8th		0x0E		Descriptor element length	Current ground
		0x05		DAEL descriptor code	
		0x04		FLT data type code	
		0x08		Maximum data length	
		"Current Gnd"		Ancillary value is data element name	
9th		0x0B		Descriptor element length	Parameters that triggered breaker
		0x05		DAEL descriptor code	
		0x06		BSTR data type code	
		0x02		Maximum data length	
		"Triggers"		Ancillary value is data element name	
10th		0x0E		Descriptor element length	Fault classification
		0x05		DAEL descriptor code	
		0x01		VSTR data type code	
		0x08		Maximum data length	
		"Fault Class"		Ancillary value is data element name	
11th		0x07		Descriptor element length	1/100th kilometers distance to fault
		0x05		DAEL descriptor code	
		0x02		UINT data type code	
		0x04		Maximum data length	
		"Dist"		Ancillary value is data element name	

EX 42	Electrical fault example 3. **Table 30** shows a typical event data set returned after a fault. This data set is reported with a group 88, variation 1 object. It is an appropriate data set for the data set descriptors in Table 27 and Table 28.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 30—Electrical fault data set

Data set element		Element contents		Element description	
1st		0x02		Value length	Identifier
		0x0000		Value is data set identifier	
2rd		0x06		Value length	Time of event
		0xF5 0x7C 0x38 0x01 0xF7 0x00		Value is time when event occurred	
3rd		0x04		Value length	Current phase A
		1328.98		Value	
4th		0x04		Value length	Current phase B
		420.17		Value	
5th		0x04		Value length	Current phase C
		1048.43		Value	
6th		0x04		Value length	Current ground
		342.95		Value	
7th		0x01		Value length	Parameters that triggered breaker
		0x0B		Value	
8th		0x03		Value length	Fault classification
		"ACG"		Value	
9th		0x01		Value length	1/100th kM distance to fault
		141		Value	

5.4.13 Example Messages

EX 43	This is an example of the master reading all of the data set descriptor in an outstation, and the outstation returning a single descriptor described in Table 27.

►►► Request Message

C3	01	56	01	06
AC	FC	Grp	Var	Qual

◄◄◄ Response Message (beginning)

C3	81	00	00	56	01	5B	01	7D	00	05
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Size prefix		← len

◄◄◄ Continuation of Response Message

01	00	00	00	00	11	04	00	00	46	64
ID	type	max	identifier →		← len	NAME	none	max	F	d

◄◄◄ Continuation of Response Message

72	20	31	31	2D	41	20	46	61	75	6C
r		1	1	–	A		F	a	u	l

◄◄◄ Continuation of Response Message

74	10	05	04	08	43	75	72	72	65	6E
t →	← len	DAEL	FLT	max	C	u	r	r	e	n

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

◄◄◄ Continuation of Response Message

74	20	50	68	73	20	41	10	05	04	08
t		P	h	s		A → |	← len	DAEL	FLT |	max

◄◄◄ Continuation of Response Message

43	75	72	72	65	6E	74	20	50	68	73
C	u	r	r	e	n	t		P	h	s

◄◄◄ Continuation of Response Message

20	42	10	05	04	08	43	75	72	72	65
	B →	← len	DAEL	FLT	max	C	u	r	r	e

◄◄◄ Continuation of Response Message

6E	74	20	50	68	73	20	43	0E	05	04
n	t		P	h	s		C →	← len	DAEL	FLT

◄◄◄ Continuation of Response Message

08	43	75	72	72	65	6E	74	20	47	6E
max	C	u	r	r	e	n	t		G	n

◄◄◄ Continuation of Response Message

64	0B	05	06	02	54	72	69	67	67	65
d → |	← len	DAEL	BSTR	max	T	r	i	g	g	e

◄◄◄ Continuation of Response Message

72	73	0E	05	01	08	46	61	75	6C	74
r	s →	← len	DAEL	VSTR	max	F	a	u	l	t

◄◄◄ Continuation of Response Message

20	43	6C	61	73	73	07	05	02	04	44
	C	l	a	s	s → |	← len	DAEL	UINT	max	D

◄◄◄ Continuation of Response Message

69	73	74
i	s	t → |

EX 44	This is an example of an electrical fault data set event returned in a response to request for class 1, 2, and 3 events.

►►► Request Message

C8	01	3C	02	06	3C	03	06	3C	04	06
| AC |	FC |	Grp |	Var |	Qual |	Grp |	Var |	Qual |	Grp |	Var |	Qual |

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

◄◄◄ Response Message (beginning)

E8	81	00	00	58	01	5B	01	26	00	02
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Size prefix		← len

◄◄◄ Continuation of Response Message

00	00	06	F5	7C	38	01	F7	00	04	5C
ID	→	← len			time			→	← len	

◄◄◄ Continuation of Response Message

1F	A6	44	04	C3	15	D2	43	04	C3	0D
current phs A		→	← len		current phs B		→	← len		

◄◄◄ Continuation of Response Message

83	44	04	9A	79	AB	43	01	0B	03	41
current phs C →		← len		current gnd		→	← len	trig →	← len	A

◄◄◄ Continuation of Response Message

43	47	01	8D
C	G →	← len	dist →

►►► Confirm Message

C3	00
AC	FC

5.5 Device attributes

DNP3 provides a means to electronically discover various outstation features and attributes. The original purpose was to simplify the burden of configuring master stations. Ideally, when either a master or outstation starts up, the master can ask for information from the outstation and then perform self-configuration. Being able to read and write the attributes has other benefits that are beyond the scope of this document.

DNP3 provides a set of standardized device attributes and an extensible means for vendors to incorporate vendor or user-specific attributes.

5.5.1 Group 0 and attribute sets

Object group 0 is assigned for attributes that relate to the outstation device itself. Variations for object group 0 represent attributes in the outstation device.

A device may support multiple sets of device attributes. These are differentiated using index numbers. The set of standardized device attributes defined by DNP3 are accessible at index 0. Vendors or users define attributes at indexes 1 and upward for private use and applications.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5.5.2 Object variations

Attribute variation numbers begin at 255 and decrease downward. This convention is designed to permit adding attribute variations in a consistent manner to other object groups in the future.

Two special variations are designed to assist master stations in collecting the attributes supported by an outstation. Outstations that implement attribute objects shall include support for these two variations at each index corresponding to an attribute set.

— Variation 255 is used to request a list of the attribute variation numbers supported by an outstation.

— Variation 254 is used to request all of the attribute objects from an outstation in a single response.

Variations 1 through 253 are pre-assigned or reserved for assignment by the DNP User's Group.

5.5.3 Function codes

All attribute variations are readable from a master by using a read request, function code 1 (READ). Variations 254, 255, and many others cannot be written, but some variations may optionally be writable depending on the outstation capabilities. Function code 2 (WRITE) is used for the write requests.

5.5.4 General attribute object formats

Every attribute object, except variation 254, has a data type code, length, and value format. The data type code and length provide keys for master and outstation devices to interpret the value. The value is self-explanatory.

5.5.4.1 Pictorial

5.5.4.2 Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code as described in 5.5.4.3.

UINT8: Length.

Specifies the number of octets required to convey the attribute value that follows.

VARIANT: Attribute value

Encoding of the attribute value is dependent on the variation's data type code. If more than one octet is required for a numeric formatted attribute, the least significant octet is transmitted first. The number of octets in this field is specified by the length field.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5.5.4.3 Attribute data type codes

Every attribute value is constructed according to one of the attribute data type codes described in Table 31.

Table 31 —Attribute data type codes

Code	Code name	Description
1	VSTR	Visible characters suitable for print and display.
2	UINT	Unsigned integer.
3	INT	Signed integer.
4	FLT	Floating-point.
5	OSTR	Octet string.
6	BSTR	Bit string.
254	U8BS8LIST	List of UINT8–BSTR8 pairs.
255	U8BS8EXLIST	Extended list of UINT8–BSTR8 pairs. Object length is 256 plus the value in the object's length octet.

Attribute data type codes do not specify the number of octets required to convey the attribute value. The second octet in a DNP3 attribute object always specifies the number of octets. For example, an outstation would most likely report an unsigned integer with 1, 2, or 4 octets.

Floating-point types are limited to 4 bytes and 8 bytes for transporting 32-bit and 64-bit floating-point values. No other sizes are acceptable. Implementers that use 64-bit floating-point values shall carefully consider the consequences of possible non-interoperability.

The bit alignment in bit string types requires the first bit to appear in bit 0 of the first byte of the attribute value. If the number of bits in a bit string is not an integral of 8, the unused bits in the last octet of the value shall be cleared to 0.

Code 254 indicates that the data type is a list of UINT8–BSTR8 pairs and that the length octet in the object specifies the object's true length.

Code 255 indicates that the data type is an extended list of UINT8–BSTR8 pairs and that the true length of the object is 256 plus the contents of the length octet in the object.

NOTE—A list of UINT8–BSTR8 pairs contains one or more sets of a UINT8 value followed by a BSTR8 value.

5.5.5 Reading attributes

The following examples illustrate reading attributes.

EX 45	This example shows a request to read the maximum receive fragment size in the outstation.

►►► Request Message

C3	01	00	F1	00	00	00
AC	FC	Grp	Var	Qual	Range	

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

◄◄◄ Response Message (beginning)

C3	81	00	00	0	F1	00	00	00	02	02
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range		←	

◄◄◄ Continuation of Response Message

DC	05
	→

The response contains a single object. Observe that the first octet specifies the data type (2, unsigned integer); the second octet specifies the number of octets (2) in the actual attribute value (0x05DC) reported by the next two octets. The receive fragment size was reported as being 1500 octets.

5.5.6 Reading a list of attribute variations

This subclause describes reading a list of attribute variations supported in an outstation.

5.5.6.1 Variation 255

Variation 255 has special meaning. It is used to retrieve a list of attribute variation numbers supported by an outstation. This object has a variable length that depends on the count of attribute variations supported by the outstation. The list does not include variations 254 and 255.

5.5.6.2 Retrieving a list of standard attribute variations

The procedure for uploading a list of DNP3-defined device attributes from an outstation is as follows:

a) The master issues a read request, using the ***read*** function code for object group 0, variation 255, from index 0.

b) The outstation responds with a list of the device attribute variations, and the properties of those attribute variations, that it supports. The response uses qualifier code 0x17, and the 1-octet range field holds a count of 1.

5.5.6.2.1 Example

EX 46	This example shows a request to read a list of attribute variations from the outstation. The outstation in this example supports 26 different device attribute variations: 217 to 233, 237 to 241, 248 to 250 and 252.

►►► Request Message

C3	01	00	FF	00	00	00
AC	FC	Grp	Var	Qual	Range	

◄◄◄ Response Message (beginning)

C3	81	00	00	00	FF	17	01	00	FE	34
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Index Prfx	Data Type	Length

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

◄◄◄ Continuation of Response Message

D9	00	DA	00	DB	00	DC	00	DD	00	DE
Var	Prop	Var	Prop	Var	Prop	Var	Prop	Var	Prop	Var

◄◄◄ Continuation of Response Message

00	DF	00	E0	00	E1	00	E2	00	E3	00
Prop	Var	Prop	Var	Prop	Var	Prop	Var	Prop	Var	Prop

◄◄◄ Continuation of Response Message

E4	00	E5	00	E6	00	E7	00	E8	00	E9
Var	Prop	Var	Prop	Var	Prop	Var	Prop	Var	Prop	Var

◄◄◄ Continuation of Response Message

00	ED	00	EE	00	EF	00	F0	01	F1	00
Prop	Var	Prop	Var	Prop	Var	Prop	Var	Prop	Var	Prop

◄◄◄ Continuation of Response Message

F8	00	F9	00	FA	00	FC	00
Var	Prop	Var	Prop	Var	Prop	Var	Prop

Note that variations 254 (0xFE) and 255 (0xFF) are not included in the list even though the device shall support these. Also notice that attribute 240 (0xF0), Maximum Transmit Fragment Size, is writable and all others are not.

5.5.6.3 Retrieving a list of private attribute variations

The procedure for uploading a list of DNP3-defined device attributes from an outstation is as follows:

a) The master issues a read request, using the ***read*** function code for object group 0, variation 255, from the index corresponding to the attribute set, index 1 or higher.

b) The outstation responds with a list of the device attribute variations, and the properties of those attribute variations, that it supports. The response uses qualifier code 0x17, and the 1-octet range field holds a count of 1.

5.5.6.3.1 Example

EX 47	This example shows a request to read a list of all variations available in the user-specific attributes set for ABC Corp defined at index 1.

►►► Request Message

C3	01	00	FF	00	01	01
AC	FC	Grp	Var	Qual	Range	

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

◄◄◄ Response Message (beginning)

C3	81	00	00	00	FF	17	01	01	FE	0A
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Index Prfx	Data Type	Length

◄◄◄ Continuation of Response Message

F0	00	F6	00	F7	00	F8	00	F9	00
Var	Prop	Var	Prop	Var	Prop	Var	Prop	Var	Prop

The response shows that the ABC Corp set of attributes has variation numbers 240 and 246 through 249, and all of those attributes are read only. Note that variations 254 (0xFE) and 255 (0xFF) are not included in the list even though the device shall support these.

5.5.7 Reading all attributes single request

There are two methods to read all of the attributes associated with a single point.

5.5.7.1 Specific variations request

The specific variation approach requires the master to know which attributes are available first. To find out, the master sends a request for a list of attributes using variation 255 in the request as described in 5.5.6.

The second step requires the master to construct a request containing one object header in the request for each attribute it desires in the response. This method can cause a lengthy request message but does not affect the response length.

As an example, suppose an outstation supports the variations 221, 224, 229, 233, and 239 at index 0. The request would include 5 object headers:

a) Group 0, variation 0xDD, qualifier 0x00 and range 0xZZ, 0xZZ.

b) Group 0, variation 0xE0, qualifier 0x00 and range 0xZZ, 0xZZ.

c) Group 0, variation 0xE5, qualifier 0x00 and range 0xZZ, 0xZZ.

d) Group 0, variation 0xE9, qualifier 0x00 and range 0xZZ, 0xZZ.

e) Group 0, variation 0xEF, qualifier 0x00 and range 0xZZ, 0xZZ.

Where 0xZZ specifies the index corresponding to the desired attribute set.

5.5.7.2 Non-specific variations request

The non-specific variation approach requires the master to construct a request containing a single object header having variation 254:

— Group 0, variation 0xFE, qualifier 0x00 and range 0xZZ, 0xZZ (where 0xZZ specifies the index corresponding to the desired attribute set).

The master can then choose any, or all, of the attribute objects returned in the response for which it has an interest.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

5.5.8 Writing attributes

Those attributes in an outstation that are writable can be set from the master station.

To determine which of the attributes are writable, the master shall first send a request for a list of attributes using variation 255 in the request as described in 5.5.6. Only those attributes in the returned list with the writable property bit set can be written; all others cannot.

The objects in a write message are formatted according to 5.5.4.

The allowed values that may be written to numeric-valued attributes are not specified unless limits appear in the descriptions in the DNP3 Data Object Library. Vendors may limit the range to values suitable for their device. If a master writes a value that is unacceptable to the outstation, the outstation shall not change its existing attribute value, and the internal indications bit IIN2.2 [PARAMETER_ERROR] shall be set in the response.

Vendors are not required to store attribute values in non-volatile memory. Therefore, attributes that were written by the master are not guaranteed to retain their values after a reset of any kind and for any reason.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

6. Application layer—part 3

6.1 State tables and diagrams

6.1.1 Outstation fragment state table

The purpose of this state table is to specify an outstation's behavior with regard to fragment reception and transmission.

The outstation needs to examine the FIR, FIN, and UNS bits and the SEQ value in the application control octet of received fragments. It shall also inspect the Application Layer function code. In addition, it needs to remember the SEQ value and all of the octets in request fragments that it accepts and in the response fragments that it transmits.

The outstation accepts request fragments if they contain a *valid request* and meet the criteria in the table; otherwise it discards the fragments and remains in the same state. The outstation does not need to remember application control octet bits or values or any of the octets from fragments that it discards.

In this clause, the term "valid request" refers to a fragment received with a function code that has a value in the range of 1 to 128 **and** that is listed in Table 32, **and** the FIR, FIN, and UNS bits in the application control octet have the following conditions:

— FIR = 1

— FIN = 1

— UNS = 0

The outstation shall maintain several local variables for each master with which it communicates.

— FirstValidRequestAccepted. This boolean variable is used to synchronize the processing of SEQ values in valid requests. The value of this variable is cleared to false at startup, immediately following a restart. It is set to true when the first valid request is received as shown in the table.

— ECSN.[24] This variable's name stands for **E**xpected **C**onfirm **S**equence **N**umber. (The name is short so that it fits in the state table.) If the outstation is expecting a confirmation to a **solicited** response, the value of this variable contains the sequence number that appears in the application control octet of the **solicited** response awaiting confirmation; otherwise, it has a value of NE, which stands for **N**othing **E**xpected. Only **solicited** confirmations that are received with a sequence number matching ECSN are processed; all others are discarded. This variable does not have any significance for **unsolicited** responses, although its value may be set or examined while awaiting a confirmation to an unsolicited response.

— The SEQ value from its most recently accepted valid request fragment.

— The Application Layer octets from its most recently accepted valid request fragment.

— The FIR, FIN, and CON bits and the SEQ value in its most recently transmitted **solicited** response fragment.

[24] It is understood that other implementations can exist that should yield correct results. This specification uses ECSN only to explain the intended behavior.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— The Application Layer octets from its most recently transmitted **solicited** response fragment.

— The FIR, FIN, and CON bits and the SEQ value from its most recently transmitted **unsolicited** response fragment.

— The Application Layer octets from its most recently transmitted **unsolicited** response fragment.

Outstation software requires three states for proper reception.

a) **Idle** state: The software is idle waiting for a fragment to arrive or for an event to occur that would trigger an unsolicited response. In some cases, a deferred request may be available upon entry to this state as a consequence of actions in another state. When there is such a deferred request available, it shall be processed immediately as though it had just been received. The software starts up in the idle state.

b) **Wait Solicited Confirm** state (abbreviated WaitSolCfm in table): The device received from the master a request that caused the outstation to send a response requiring a confirmation (e.g., includes events or has multiple fragments), and the outstation is waiting for the confirmation.

c) **Wait Unsolicited Confirm** state (abbreviated WaitUnsolCfm in table): The device transmitted an unsolicited response and is waiting for a confirmation from the master. It should be noted that expectation of a **solicited** confirm can occur while the software is in the Wait Unsolicited Confirm state due to a non-read request being issued by the master for which the outstation is obligated to set the CON bit in the solicited response.

Keys for understanding Table 32:

— The events referred to in italics in the table heading are triggers that initiate an action and possible transition to a different state. They do not refer to the buffered events or event objects transported in messages.

— Labels inside square brackets [ ] in column B are triggering event names associated with the specific state in which it occurs. Note that the same name may be used in different states.

— X means don't care.

— == means "Is Equal To" (For example, ECSN == NE.)

— != means "Not Equal To" (For example, !=N means not equal to N.)

— The dash symbol — means "Not Applicable."

— In column B, fragments that are received are assumed to be addressed to the outstation unless "with broadcast address" is stated. Fragments having other destination addresses are ignored.

— N and M represent sequence numbers in an application control octet.

N is used with regard to the SEQ number received from a non-broadcast, valid master request, or its response.

M is used in regard to the SEQ number in the previous (most recently sent) unsolicited response transmited from the outstation.

— When "txCON = 0" or "txCON = 1" appears in columns D and E, it refers to whether the confirm bit is set to zero or one in the application control octet of the outstation's response.

— When "no response" appears in column E, it refers to the case where a request is received that does not require a response from the outstation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— Whenever a **solicited** response is transmitted that requires confirmation, it is assumed that a **solicited** confirmation timer is re-started. Whenever an **unsolicited** response is transmitted, it is assumed that an **unsolicited** confirmation timer is re-started. These timers are different from each other and may need to run concurrently.

— Upon entering the Idle state, if a deferred read request exists, it is handled as though it were received immediately after entering the Idle state.

— For each row of Table 32, the software should behave as follows:

The software is in the state shown in column A and waits until something occurs that initiates further action.

Column B describes the cause for action.

Column C shows the UNS bit, SEQ number, and Function Code in the received fragment when an action is triggered by receipt of a fragment from the master or is associated with a deferred request.

Column D describes the action performed.

After performing the action in column D, the software transitions to the state specified in column E. Column E specifies one of the states listed under column A.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 32—Outstation fragment state table

Current state	Event that triggers an action and possible transition				Action		Transition to state	
A	B	C			D		E	
If the software state is	and this occurs	and received fragment contains			then perform this action		and go to this state	
		UNS	SEQ	Function Code				
Idle	[RESTART] A restart occurred and outstation is configured to send unsolicited responses.	—	—	—	Send unsolicited NULL response with UNS, CON and IIN1.7 bits set and seq = any legal value.		WaitUnsolCfm	1
	[DEFERRED_READ_EXISTS] A read request exists that was deferred in another state.	—	—	—	Process read request and send response if required. If txCON set to 1, then set ECSN to sequence number in transmitted fragment.		If txCON = 0, Idle; else WaitSolCfm	2
	[UNSOL_TRIGGER] The outstation is configured to send unsolicited responses, ECSN == NE and something occurs to initiate a new original unsolicited response sequence.	—	—	—	Increment M, modulo 16, and send unsolicited response with UNS and CON bits set.		WaitUnsolCfm	3
	[BROADCAST_FRAG_RCVD] Fragment received with broacast address.	0	X	Valid Request	Accept fragment and process request. Never send a response.		Idle	4
	[FIRST_FRAG_RCVD] First request fragment received following a restart.	0	X	Valid Request	Accept fragment and process request. Set variable FirstValidRequestAccepted to true. Send response if required. If txCON set to 1, then set ECSN to sequence number in transmitted fragment.		If no response or txCON = 0, Idle; else WaitSolCfm	5
	[NEW_FRAG_RCVD] Request fragment received.	0	!=N	Valid Request	Accept fragment and process request. Send response if required. If txCON set to 1, then set ECSN to sequence number in transmitted fragment.		If no response or txCON = 0, Idle; else WaitSolCfm	6
	[REPEAT_FRAG_RCVD] Request fragment received.	0	N	Valid Request	Compare octet-by-octet with previous request fragment. Do they match?		—	7
					Yes	Accept fragment then send same response. Do not process the request.	Idle	8

Copyright © 2010 IEEE. All rights reserved.

Current state	Event that triggers an action and possible transition				Action		Transition to state	
A	B	C			D		E	
If the software state is	and this occurs	and received fragment contains			then perform this action		and go to this state	
		UNS	SEQ	Function Code				
					No	Accept fragment and process request. Send response if required. If txCON set to 1, then set ECSN to sequence number in transmitted fragment.	If no response or txCON = 0, Idle; else WaitSolCfm	9
	[CONFIRM_RCVD] Confirm received.	X	X	Confirm	Discard confirm fragment; do not remove any events.		Idle	11
Wait Solicited Confirm	[MATCHING_SOL_CONFIRM_RCVD] Confirm fragment received. (N is SEQ value sent in response awaiting confirm.)	0	== ECSN	Confirm	Accept fragment, set ECSN to NE, and process confirm. If one fragment of multi-fragment message is being confirmed, and a subsequent fragment is necessary, then increment N, modulo 16, and send the next fragment. If txCON set to 1, then set ECSN to sequence number in transmitted fragment.		If fragment is transmitted and txCON = 1, WaitSolCfm; else Idle	11
	[NON-MATCHING_SOL_CONFIRM_RCVD] Confirm fragment received. (N is SEQ value sent in response awaiting confirm.)	0	!= ECSN	Confirm	Discard confirm fragment; do not remove any events.		WaitSolCfm	12
	[UNSOL_CFM_RCVD] Unsolicited confirm received.	1	X	Confirm	Discard confirm fragment; do not remove any events.		WaitSolCfm	13
	[BROADCAST_FRAG_RCVD] Fragment received with broadcast address.	0	X	Valid Request	Assume confirmation failed and set ECSN to NE. Accept fragment and process request. Never send a response.		Idle	14
	[NEW_FRAG_RCVD] Request fragment received. (N is SEQ value in last valid request received.)	0	!=N	Valid Request	Assume confirmation failed and set ECSN to NE. Accept fragment and process request. Send response if required. If txCON set to 1, then set ECSN to sequence number in transmitted fragment.		If no response or txCON = 0, Idle; else WaitSolCfm	15
	[REPEAT_FRAG_RCVD] Request fragment received. (N is SEQ value in last valid request received.)	0	N	Valid Request	Compare octet-by-octet with previous request fragment. Do they match?		—	16
					Yes	Accept fragment then send same response and restart confirmation timer. Do not process the request.	WaitSolCfm	17
					No	Assume confirmation failed and set ECSN to NE. Accept fragment and process request. Send response if required. If txCON set to 1, then set ECSN to sequence number in transmitted fragment.	If no response or txCON = 0, Idle; else WaitSolCfm	18

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Current state	Event that triggers an action and possible transition				Action		Transition to state	
A	B	C			D		E	
If the software state is	and this occurs	and received fragment contains			then perform this action		and go to this state	
		UNS	SEQ	Function Code				
	[SOL_CONFIRM_TIMOUT] Timeout of solicited confirmation timer.	—	—	—	Note failure to confirm and set ECSN to NE. Do not remove any events, do not send retries.		Idle	19
Wait Unsolicited Confirm	[MATCHING_SOL_CONFIRM_RCVD] Solicited confirm received, seq number == ECSN.	0	== ECSN	Confirm	Process the confirmation and set ECSN to NE.		WaitUnsolCfm	20
	[NON-MATCHING_SOL_CONFIRM_RCVD] Solicited confirm received, seq number != ECSN.	0	!= ECSN	Confirm	Discard confirm fragment; do not remove any events.		WaitUnsolCfm	21
	[MATCHING_UNSOL_CFM_RCVD] Unsol confirm received.	1	M	Confirm	Accept fragment and process confirm.		If ECSN == NE, Idle; else WaitSolCfm	22
	[NON-MATCHING_UNSOL_CFM_RCVD] Confirm fragment received.	1	!=M	Confirm	Discard confirm fragment.		WaitUnsolCfm	23
	[BROADCAST_FRAG_RCVD] Fragment received with broadcast address.	0	X	Valid Request	Set ECSN to NE. If a deferred read request exists, discard it. Accept fragment and process request. Never send a response.		WaitUnsolCfm	24
	[READ_REQ_RCVD] Read request received.	0	X	Read Request	Set ECSN to NE. If a deferred request already exists, replace it with this request. Defer building the read response by holding request until the software enters the Idle state.		WaitUnsolCfm	25
	[NON-RD_REQ_RCVD] A non-read request is received.	0	!= N	Non-read Valid Request	Set ECSN to NE. If a deferred read request exists, discard it. Accept fragment and send response. If txCON set to 1, then set ECSN to sequence number in transmitted fragment.		WaitUnsolCfm	26
	[REPEAT_NON-RD_ RCVD] A non-read request is received.	0	N	Non-read Valid Request	If a deferred read request exists, discard it. Compare octet-by-octet with previous request. Do they match?		—	27
					Yes	Accept fragment then send same response. Do not process the request.	WaitUnsolCfm	28
					No	Set ECSN to NE. Accept fragment and process request. Send response if required. If txCON set to 1, then set ECSN to sequence number in transmitted fragment.	WaitUnsolCfm	29

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Current state	Event that triggers an action and possible transition				Action		Transition to state	
A	B	C			D		E	
If the software state is	and this occurs	and received fragment contains			then perform this action		and go to this state	
		UNS	SEQ	Function Code				
	[SOL_CONFIRM_TIMOUT] Timeout of solicited confirmation timer.	—	—	—	Set ECSN to NE. Do not remove any events, do not send retries.		WaitUnsolCfm	30
	[UNSOL_CONFIRM_TIMOUT_ DEFER] Timeout waiting for confirm. A read request is deferred.	—	—	—	Note failure to confirm and terminate expectation of an unsolicited confirm. Terminate unsolicited response series.		Idle	31
	[UNSOL_CONFIRM_TIMOUT_NO_DEFER] Timeout waiting for confirm. A read request is not deferred.	—	—	—	Note failure to confirm. Have configured number of unsolicited retries been transmitted?		—	32
					Yes	Assume confirmation failed and terminate expectation of an unsolicited confirm. Terminate unsolicited response series.	If ECSN == NE, Idle; else WaitSolCfm	33
					No	Transmit an identical or regenerated retry unsolicited response.	WaitUnsolCfm	34

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

6.1.2 Outstation fragment state diagram

Figure 25 diagrams the outstation fragment state.

Figure 25—Outstation fragment state diagram

6.1.3 Master solicited response reception state table

The purpose of this reception state table is to specify a master's behavior when fragments are received with the UNS bit equal to 0 in the application control octet. The master software needs to examine the FIR and FIN bits and the SEQ value in the application control octet. It also needs to remember the FIR and FIN bits, the SEQ value, and all of the octets from the previously accepted solicited response fragment. The master accepts the fragment if it meets the criteria in the table; otherwise it discards the fragment. The master does not need to remember application control octet bits or values or any of the octets from fragments that it discards. The master shall also remember the SEQ value from the last request fragment that it sent in a request to the outstation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

NOTE

A master should keep a separate set of variables for **each** outstation with which it communicates.

The variables include as a minimum

— The FIR and FIN bits and the SEQ value from the most recently accepted solicited response fragment.

— The octets from the most recently accepted solicited response fragment.

— The SEQ value in the most recent request fragment it transmitted.

The master maintains a separate and independent set of variables for solicited and unsolicited responses. The state machines should run concurrently in masters that support unsolicited responses.

An outstation should also maintain additional information from the Transport Function (Clause 8) and from the Data Link Layer (Clause 9).

Master software for handling solicited responses requires three states for proper operation.

a) **Idle** state: The master is waiting for the DNP3 user software at a higher layer to initiate a request. The master starts in this state immediately following a reset.

b) **AwaitFirst** state: The software is waiting for the first fragment of the expected response to arrive from the outstation.

c) **Assembly** state: While in this state, the master is awaiting more fragments from a multi-fragment response.

Keys for understanding Table 33:

— The events referred to in italics in the table heading are triggers that initiate an action and possible transition to a different state. They do not refer to the buffered events or event objects transported in messages.

— X means don't care.

— The dash symbol — means "Not Applicable."

— N is any valid sequence number.

— N + 1 is N plus 1 modulo 16.

— != means "Not Equal To" (For example, !=N means not equal to N.)

— For each row of the Table 33, the software should behave as follows:

If the software is currently in the state listed in column A,

and the user initiates transmission of a request, the response timer times out or a fragment is received with the fields of the application control octet as shown in column B,

and the fields of the application control octet in the most recently accepted solicited fragment were as shown in column C,

and the sequence number in the request's application control octet was as shown in column D,

then perform the actions described in column E,

and after performing the action in column E, transition to the state specified column F; column F specifies one of the states listed under column A.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 33—Master reception state table, solicited responses

Current state	Event that triggers an action and possible transition							Action	Transition to state	
A	B			C			D	E	F	
If the software state is	and the user initiates a new request, the response timer times out or a fragment is received with these fields			and the fields in the most recently **accepted solicited** fragment were			and field in request was	then perform this action	and go to this state	
	FIR	FIN	SEQ	FIR	FIN	SEQ	SEQ			
Idle	X	X	X	X	X	X	X	Discard fragment and do not confirm.	Idle	1
	User initiates a request			—	—	—	—	Transmit the user request and if response expected, start fragment receive timer.	If response expected AwaitFirst; else Idle.	2
AwaitFirst	0	X	X	X	X	X	X	Discard fragment and do not confirm.	AwaitFirst	3
	1	0	N	X	X	X	N	Send confirm if requested, accept fragment, process fragment, and start fragment receive timer.	Assembly	4
	1	X	!=N	X	X	X	N	Discard fragment and do not confirm.	Idle	5
	1	1	N	X	X	X	N	Send confirm if requested, accept fragment, and process fragment.	Idle	6
	Response timer times out			X	X	X	X	Note lack of response.	Idle	7
Assembly	0	0	N	0	0	N	X	Compare octet-by-octet with previous accepted fragment. If octets match, send confirm if requested, take no further action, and start fragment receive timer. If octets do not match, discard fragment and do not confirm.	If octets match, Assembly; else Idle	8
	0	0	N	1	0	N	X	Discard fragment and do not confirm.	Idle	9
	0	0	N + 1	X	0	N	X	Send confirm if requested, accept fragment, process fragment, and start fragment receive timer.	Assembly	10
	0	0	!=N and != N + 1	X	0	N	X	Discard fragment and do not confirm.	Idle	11
	0	1	N + 1	X	0	N	X	Send confirm if requested, accept fragment, and process fragment.	Idle	12
	0	1	!= N + 1	X	0	N	X	Discard fragment and do not confirm.	Idle	13

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Current state	Event that triggers an action and possible transition							Action	Transition to state	
A	B			C			D	E	F	
If the software state is	and the user initiates a new request, the response timer times out or a fragment is received with these fields			and the fields in the most recently **accepted solicited** fragment were			and field in request was	then perform this action	and go to this state	
	FIR	FIN	SEQ	FIR	FIN	SEQ	SEQ			
	1	0	N	1	0	N	N	Compare octet-by-octet with previous accepted fragment. If octets match, send confirm if requested, take no further action, and start fragment receive timer. If octets do not match, discard fragment and do not confirm.	If octets match, Assembly; else Idle	14
	1	0	N	0	0	N	X	Discard fragment and do not confirm.	Idle	15
	1	0	!=N	X	0	N	X	Discard fragment and do not confirm.	Idle	16
	1	1	X	X	X	X	X	Discard fragment and do not confirm.	Idle	17
	Response timer times out			X	X	X	X	Note lack of response.	Idle	18

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

6.1.4 Master solicited response reception state diagram

Figure 26 diagrams the master solicited response reception state.

Figure 26—Master solicited response reception state diagram

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

6.1.5 Master unsolicited response reception state table

The purpose of this reception state table is to specify a master's behavior when fragments are received with the UNS bit equal to 1 in the application control octet. The master software needs to examine the SEQ value in the application control octet. It also needs to remember the SEQ value and all of the octets from the previously accepted unsolicited response fragment.

NOTE

A master should keep a separate set of variables for **each** outstation with which it communicates.

The variables include as a minimum

— The SEQ value from the most recently accepted unsolicited response fragment.

— The octets from the most recently accepted unsolicited response fragment.

The master maintains a separate and independent set of variables for solicited and unsolicited responses. The state machines should run concurrently in masters that support unsolicited responses.

An outstation should also maintain additional information from the Transport Function (Clause 8) and from the Data Link Layer (Clause 9).

Master software for handling unsolicited responses requires three states for proper operation.

a) **Startup** state: The master just started after a reset for any reason and has **not** performed an initial integrity poll. Refer to Clause 5 for more details.

b) **FirstUR** state: The master started up, completed an initial integrity poll, and is waiting for the first unsolicited response from the outstation.

c) **Idle** state: The software is idle waiting for a unsolicited response fragment to arrive.

Keys for understanding Table 34:

— ‡ means possible data loss or duplicate data.

— X means don't care.

— The dash symbol — means "Not Applicable."

— N is any valid sequence number.

— N + 1 is N plus 1 modulo 16.

— != means "Not Equal To." (For example, !=N means not equal to N.)

— IIN1.7 means IIN1.7 [DEVICE_RESTART] bit is set in the response.

— For each row of the Table 34, the software should behave as follows:

If the software is currently in the state shown in column A,

and the SEQ number in the application control octet in the received fragment is as appears in column B,

and SEQ number application control octet in most recently accepted fragment is as shown in column C,

then perform the actions stated in column D,

and go into the software state specified in column E.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 34—Master reception state table, unsolicited responses

Current state	Event that triggers an action and possible transition		Action	Transition to state	
A	B	C	D	E	
If the software state is	and a fragment is received with SEQ number	and the SEQ number in the most recently **accepted unsolicited** fragment was	then perform this action	and go to this state	
Startup	X	—	Discard fragment and do not confirm.	Startup	1
	Initial integrity poll completed	—	Prepare to receive unsolicited responses.	FirstUR	2
FirstUR	X	—	Send confirm if requested, accept fragment, and process fragment.	Idle	3
	X and IIN1.7	—	Send confirm if requested, accept fragment, process fragment, and send reset IIN1.7 request and perform integrity poll.	Idle	4
Idle	N	N	Send confirm if requested. Compare octet-by-octet with previous fragment. If octets match, take no further action. If octets do not match, accept fragment, process fragment, and perform integrity poll.	Idle	5
	N + 1	N	Send confirm if requested, accept fragment, and process fragment	Idle	6
	!= (N + 1)	N	Send confirm if requested, accept fragment, process fragment ‡, and optionally perform integrity poll.	Idle	7
	X and IIN1.7	N	Send confirm if requested, accept fragment, process fragment, and send reset IIN1.7 request and perform integrity poll.	Idle	8

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

6.1.6 Master unsolicited response reception state diagram

Figure 27 diagrams the master unsolicited response reception state.

Figure 27—Master unsolicited response reception state diagram

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7. Secure authentication

7.1 Introduction

7.1.1 Purpose

This clause defines a protocol mechanism that:

— A DNP3 outstation can use to unambiguously determine it is communicating with a user who is authorized to access the services of the outstation.

— A DNP3 master can use to unambiguously determine that it is communicating with the correct outstation.

7.1.2 References

This specification is fundamentally based on IEC/TS 62351-5 and is intended to be compliant with that specification as well as selected portions of ISO/IEC 9798-4. If there are disagreements between this specification and those references, this specification shall be considered to be correct.

The following statement is on file with the Internet Engineering Task Force (IETF) as Intellectual Property Rights (IPR) statement number 858, dated 2007-06-20. It is listed as having been submitted by "The United States of America as represented by the National Security Agency":

The U.S. Government holds U.S. Patent 6,829,355 on the "Device for and method of one-way cryptographic hashing," which has been incorporated into Federal Information Processing Standard (FIPS) 180-2 [B2].[25]

This patent was issued on December 7, 2004. The National Security Agency has made U.S. Patent 6,829,355 available royalty-free.[26]

7.1.3 Threats addressed

This specification shall address only the following security threats, as defined in IEC/TS 62351-2:

— Spoofing

— Modification

— Replay

— Eavesdropping—on exchanges of cryptographic keys only, not on other data.

— Non-repudiation—to the extent of identifying individual users of the system

[25] The numbers in brackets correspond to those of the bibliography in Annex E.
[26] At the time of writing, the URL for this document was https://datatracker.ietf.org/ipr/858/.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.1.4 General principles

This subclause describes the guiding principles behind this specification, based on the identified threats.

7.1.4.1 Authentication only

This specification addresses authentication only, not encryption or other security measures. It does not rule out the possibility of such measures being added to DNP3 later or through the use of external measures such as "bump in the wire" link encryptors.

7.1.4.2 Application layer only

This specification describes authentication at the Application Layer. Application layer authentication is necessary because:

— DNP3 may be used over a variety of different physical networks and may be "bridged" from one to the other, as in the case of a TCP/IP terminal server or IP radio. Only authentication at the Application Layer shall be used to provide end-to-end security.

— Application layer authentication permits the possibility of protection against "rogue applications" that may be co-resident with the DNP3 application and attempt to use the DNP3 link without authorization.

— Application layer authentication permits the possibility of authenticating individual users, as discussed in 7.1.4.9.

7.1.4.3 Bi-directional

This specification describes a mechanism that can be used in either transmission direction, master-to-outstation (controlling direction), or outstation-to-master (monitoring direction).

7.1.4.4 Challenge-response

The mechanism described in this specification is based on the common security concept of challenge and response. This principle has been applied for the following reasons:

— It places the responsibility for security on the device that requires authentication, which is more practical in a diverse network such as those found in the utility industry.

— It permits some communication to be left unsecured if desired, reducing bandwidth and processing requirements.

— It works effectively in a non-connection-oriented environment.

Because "response" is a keyword in DNP3, the term used in this specification is "reply."

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.1.4.5 Pre-shared keys

This specification permits pre-shared keys to be used by default. This principle recognizes the fact that many utilities may choose not to manage security credentials in a more sophisticated manner but nevertheless require some level of protection.

This specification does not prevent the use of public-key cryptography mechanisms to be used to change or distribute the pre-shared keys described here. Future specifications may standardize such methods, but they are currently out of the scope of this specification.

7.1.4.6 Backwards tolerance

This specification recommends that the following conditions be satisfied when a secure device (one implementing this authentication mechanism) communicates with a non-secure device:

— The secure device shall be able to detect that the non-secure device does not support the authentication mechanism.

— The non-secure device shall continue to operate normally after being contacted by the secure device. In other words, the authentication message cannot cause the non-secure device to fail.

— The two devices shall be able to continue to exchange information that is not considered critical.

However, the mechanism's ability to meet these conditions is largely dependent on the quality of the implementation on any particular device. This specification therefore recommends that secure devices avoid sending security messages if it is not known whether the remote device supports security.

7.1.4.7 Upgradeable

This specification permits system administrators to change algorithms, key lengths, and other security parameters to deal with future requirements. In keeping with the principle of backward tolerance, it also permits one end of a link to be upgraded at a time.

7.1.4.8 Perfect forward secrecy

This specification follows the security principle of perfect forward secrecy, as defined in IEC/TS 62351-2. If a session key is compromised, this mechanism only puts data from that particular session at risk, and it does not permit an attacker to authenticate data in future sessions.

7.1.4.9 Multiple users

This specification assumes that there may be multiple users of the system located at the site of the master. It provides a method to authenticate each of the users separately from each other and from the master itself.

The intent of this principle is to permit the outstation to conclusively identify the individual user (not just the device) that transmits any protocol message. This information can be used in audit logs for non-repudiation purposes, although such use is out of the scope of this document.

This standard allows for the possibility that outstations may limit access to certain functions, either based on the individual identities of users or based on the "roles" the users perform. However, it does not specify a mechanism for defining such user-based or role-based access control.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.2 Description of operation

This subclause describes the operation of the authentication mechanism in general terms for the benefit of first-time readers. In the case of disagreements between this subclause and 7.3, 7.3 shall be taken as correct.

7.2.1 Narrative description

This subclause describes the operation of the authentication mechanism as a text narrative.

The assumed implementation architecture of this mechanism is shown in Figure 28. Multiple users may either send unauthenticated DNP3 messages, or they may choose to authenticate selected messages. The authentication messages have the ability to distinguish between users, while normal DNP3 messages do not. The authentication messages are formatted as additional DNP3 function codes and object variations.

The software architecture used to parse, process, and distinguish between normal messages and security messages is beyond the scope of this document.

Implementers should note that logging and auditing of security events such as authentication failures is a critical part of information security. It is recommended that all implementations at a minimum log all successful and unsuccessful authentications and key changes, including the time, the DNP3 addresses, and the affected user. However, since logging is not a part of the protocol itself, the events logged and the format of the log are not parts of this standard.

Figure 28—Assumed implementation architecture

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.2.1.1 Basic concepts

The authentication mechanism is based on two concepts:

— A challenge and response protocol, as discussed in 7.1.4.4. The general mechanism is illustrated in Figure 29. Because "response" is a keyword in DNP3, the term used here is "reply."

— The concept of a Keyed-Hash Message Authentication Code (HMAC) that both the outstations and the masters calculate based on each Application Service Data Unit (ASDU, or protocol message) that is to be authenticated.

An HMAC algorithm is a mathematical calculation that takes a protocol message as input, produces a smaller piece of data as output, and has the following characteristics:

— The value of the output is sensitive to small changes in the input message, so the output of the HMAC can be used to detect whether the message was modified.

— The calculation makes intrinsic use of a secret key that is shared by both ends of the communication.

— It is extremely difficult to determine the secret key by viewing the HMAC output.

— It is nearly impossible to determine the original message from the HMAC.

— It is difficult to find two messages that produce the same HMAC.

— This challenge-response mechanism using an HMAC is a "unilateral, two-pass authentication" mechanism as described in ISO/IEC 9798-4.

7.2.1.2 Initiating the challenge

The challenge may be initiated either by the master or the outstation.

Devices shall issue challenges to protect specific ASDUs that the device considers to be critical. The challenger issues the challenge immediately after receiving the critical ASDU, before taking any action on it.

Outstations shall consider all output operations (controls, setpoint adjustments, parameter settings, etc.) to be critical. Other mandatory critical operations are described in 7.3.2.1.2. Each implementation may define additional mandatory critical operations.

To protect against replay attacks, the challenge message contains data that changes randomly each time a challenge is issued.

The challenger specifies in the challenge message the cryptographic Keyed-Hash Message Authentication code (HMAC) algorithm for the responder to use when building the response.

7.2.1.3 Replying to the challenge

The device (either master or outstation) that receives the challenge shall reply before communications can continue.

The responder performs the HMAC algorithm specified in the challenge message to produce the reply. A shared Session Key known to both devices is an integral part of the computation. The following types of information are included in the computation:

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— A number specifying the user on the master side is included.

— The challenge data is included, to protect against replay attacks.

— If the challenger is protecting a specific critical ASDU, data from that ASDU is also included in the computation. This protects against modification of the ASDU by an attacker.

The reply includes the resulting HMAC value.

7.2.1.4 Authenticating

Upon receiving the reply, the challenger performs the same calculation on the same data used by the responder. If the results match, the challenger permits communications to continue. If the challenger was protecting a particular ASDU, it processes the ASDU.

7.2.1.5 Authentication failure

If the authentication fails, the challenger shall not use data from the challenged message. If the challenger is an outstation, it shall not perform the operation requested by the master. The challenger may then choose to transmit an error message. To help protect against denial-of-service attacks and attackers learning from repeated challenges, the challenger shall cease to transmit error messages after a configurable number of failures. Refer to 7.4.1.4.6 for more details about the configurable maximum error count.

7.2.1.6 Aggressive mode

To reduce bandwidth usage, a responder attempting a critical operation may optionally "anticipate" the challenge and send the HMAC value in the same ASDU being protected. This practice is known as "aggressive mode." It eliminates the challenge and reply messages. Aggressive mode provides a lower level of security because not as much data is changing in each message.

For this reason, aggressive mode is optional in IEC/TS 62351-5. However, the value of aggressive mode is considered high enough for DNP3 that all DNP3 implementations of this authentication mechanism are required to support it. Per IEC/TS 62351-5, however, all DNP3 implementations are also required to permit it to be disabled by configuration, so that individual projects can use only challenge and reply if they choose.

Aggressive mode is a "unilateral, one-pass authentication" mechanism as described in ISO/IEC 9798-4. However, it is somewhat more secure against replay attacks than the mechanism described there because the Aggressive Mode Request includes information from the most recently received challenge in addition to the sequence number required by ISO/IEC 9798-4.

7.2.1.7 Changing keys

Table 35 summarizes how cryptographic keys are used and updated in this authentication mechanism.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 35—Summary of keys used

Type	Use	Change mechanism	Range of expected change interval
Monitoring Direction Session Key	Used to authenticate data transmitted in the monitoring direction by the outstation	The master encrypts the Session Key in a Key Change message using the Update Key.	Minutes up to weeks (for infrequently communicating systems)
Control Direction Session Key	Used to authenticate data transmitted in the control direction by the master	The master encrypts the Session Key in a Key Change message using the Update Key.	Minutes up to weeks
Update Key	The master shall use the Update Key to periodically change the Session Keys	The Update Key is pre-shared by the two devices and is changed only by means external to the protocol.	Months or Years

7.2.1.7.1 Managing session keys

The Session Keys that each device uses to hash the challenge data are the most frequently used keys. A different Session Key is used in each direction, so that if the key for one direction is compromised, it does not compromise communications in the other direction. There is a different set of Session Keys and a different Update Key for each user at the master end, identified by a User Number.

The master initializes the Session Keys immediately after communications is established and regularly changes the Session Keys thereafter. This practice of periodically changing the Session Keys protects them from being compromised through analysis of the communications link.

The master uses a second key, called the Update Key, to encrypt the new Session Keys, together with the challenge data, inside a Key Change message. The use of a second key permits the master to change the Session Key even if the original Session Key was compromised. Both the Session Keys and the Update Key are symmetric keys.

The sequence for changing the Session Keys is shown in Figure 33 and Figure 34. Like the normal authentication mechanism, it is also based on challenge and reply:

— The master sends a Key Status Request message, which contains no data but serves to initiate the process. It does include a User Number that indicates the particular Update Key and set of Session Keys being queried.

— The outstation replies with a Key Status message containing the current status of the keys and some challenge data.

— The master updates the Session Keys with a Key Change message. Besides changing the keys, the Key Change message also constitutes a reply to the challenge and permits the outstation to authenticate that the correct entity is attempting to change the Session Keys.

— The outstation replies with a new Key Status message. This Key Status message indicates whether the Key Change was successful (i.e., properly received and authentic) and includes freshly generated challenge data.

— Thereafter, the master can send another Key Change message at any time, replying to the most recent challenge data it received.

The algorithm used to encrypt both the Session Keys together with the challenge data is known as a “key wrap” algorithm. The minimum required key wrap algorithms are specified in 7.4.1.2.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

If either device determines that the communications between them has failed, it shall assume the most recent set of Session Keys have been compromised and shall refuse to use them to authenticate any further Challenge or Aggressive Mode Request messages. The master shall send a Key Status Request at the earliest opportunity after detecting the communications failure, and re-initialize the Session Keys.

7.2.1.7.2 Managing update keys

As discussed in 7.1.4.9, this authentication mechanism permits multiple users of the system to be authenticated separately from the master itself. Each user is identified by his or her own User Number and has his or her own Update Key and set of Session Keys.

Each user's Update Key is rarely changed. The reason for such a change is dependent on the security policy of the organization, but it may include the Update Key being compromised, or a user leaving the organization. In such cases, the Update Key shall be changed through a mechanism external to the protocol. This mechanism is outside the scope of this document, but the Update Key shall be kept secret and cannot be obtained by eavesdropping in transit.

It is furthermore vital for security that each device keep Update Keys secret. The mechanism used to do so is out of the scope of this document, but implementers should note that if Update Keys are entered or stored on the device in an insecure fashion, the entire authentication mechanism is compromised. It is the responsibility of each master to verify that users are personally authenticated and securely associated with the Update Keys used to identify them.

7.2.2 Example message sequences

7.2.2.1 Overview of subclause

This subclause contains diagrams illustrating examples of how the authentication mechanism shall behave. This subclause is informative only. Refer to 7.3 for a formal description of the mechanism. Bold arrows in these diagrams represent authentication-specific messages.

7.2.2.2 Challenge of a critical ASDU

Figure 29 and Figure 30 illustrate the challenge and reply to a critical ASDU.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 29—Example of successful challenge of critical ASDU

Figure 30—Example of failed challenge of critical ASDU

7.2.2.3 Aggressive mode

Figure 31 and Figure 32 illustrate authentication of a critical ASDU using aggressive mode.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 31—Example of a successful aggressive mode request

Figure 32—Example of a failed aggressive mode request

7.2.2.4 Initializing and changing Session Keys

Figure 33 and Figure 34 illustrate how the master initializes and changes the Session Keys on startup, periodically, and after a communications failure.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 33—Example of Session Key initialization and periodic update

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 34—Example of communications failure followed by Session Key change

7.2.3 State machine overview

Figure 35 and Figure 36 show the major state transitions for the protocol. These diagrams are not normative, nor are they comprehensive. However, these figures *are* intended to show the general operation of the authentication protocol.

The details of the state machines are specified in 7.3. If these diagrams differ from 7.3, that subclause shall be considered to be correct.

The *Security Idle* and *Wait for Reply* states are common to both masters and outstations. The other states are specific to each type of device.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 35—Major state transitions for master

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 36—Major state transitions for outstation

7.3 Formal specification

This subclause formally describes the protocol used for this authentication mechanism. If this subclause differs from 7.2, this subclause shall be considered to be definitive.

7.3.1 Message definitions

This subclause describes the DNP3 messages used to implement the authentication mechanism. The DNP3 objects used are defined in the Data Object Library.

7.3.1.1 Master authentication implementation

DNP3 masters shall implement the authentication mechanism using the function codes and objects described in Table 36. The first column of the table shows how these function codes and objects correspond to the IEC/TS 62351-5 specification and the state machines in 7.3.2.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 36—DNP3 master messages with correlation to IEC/TS 62351-5

IEC/TS 62351-5 message	Description	Message from master contains		Outstation responds with	
		DNP3 function codes	DNP3 objects	DNP3 function codes	DNP3 objects
Challenge	Requests authentication of the preceding outstation DNP3 Response or Unsolicited Response	0x20 Authentication Request	g120v1 Authentication Challenge object	0x83 Authentication Response	g120v2 Authentication Reply object
Reply	Provides authentication of a Challenge from the outstation	0x20 Authentication Request	g120v2 Authentication Reply object	0x81 Response If authentication was successful	Whatever objects are appropriate for the normal response to the master request that caused the outstation to issue the challenge
				0x83 Authentication Response If authentication failed	g120v7 Authentication Error object
Aggressive Mode Request	Provides authentication for the current DNP3 Request	Whatever function code is in the DNP3 Request	g120v3 Authentication Aggressive Mode Request object. Shall be first object. • • • Objects appropriate for standard DNP3 request • • • g120v9 Authentication HMAC object. Shall be last object	0x81 Response If authentication was successful	Whatever objects are appropriate for the normal response to the master request
				0x83 Authentication Response If authentication failed	g120v7 Authentication Error object
Key Status Request	Requests the current status of the Session Keys	0x20 Authentication Request	g120v4 Session Key Status Request object	0x83 Authentication Response	g120v5 Session Key Status object
Key Change	Changes the symmetric Session Keys subsequently used by master and outstation for authentication	0x20 Authentication Request	g120v6 Session Key Change object	0x83 Authentication Response If authentication was successful	g120v5 Session Key Status object
				0x83 Authentication Response If authentication failed	g120v7 Authentication Error object
Error	Indicates the authentication provided in the challenge reply from outstation was incorrect or that the outstation's aggressive mode DNP3 response did not correctly authenticate	0x21 Authentication Request—No Ack	g120v7 Authentication Error object	None	None
NOTE—Shaded cells indicate optional messages.					

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.3.1.2 Outstation authentication implementation

DNP3 outstations shall implement the authentication mechanism using the function codes, objects, and Internal Indications described in Table 37. The first column shows how they correspond to the IEC/TS 62351-5 specification and the state machines in 7.3.2.

Table 37—DNP3 outstation messages with correlation to IEC/TS 62351-5

IEC/TS 62351-5 message	Description	Message from outstation contains		Message initiated because master sent	
		DNP3 function codes	DNP3 objects	DNP3 function codes	DNP3 objects
Challenge	Requests authentication of the preceding master DNP3 Request	0x83 Authentication Response	g120v1 Authentication Challenge object	Any valid function code	Whatever is appropriate to a solicited request
Reply	Provides authentication of a Challenge from the master	0x83 Authentication Response	g120v2 Authentication Reply object	0x20 Authentication Request	g120v1 Authentication Challenge object
Aggressive Mode Request	Provides authentication for the outstation's current DNP3 Response	0x81 Response or 0x82 Unsolicited Response	g120v3 Authentication Aggressive Mode Request object. Shall be first object. • • • Objects appropriate for standard DNP3 response • • • g120v9 Authentication HMAC object. Shall be last object	Any valid request function code, or Master may not have sent anything but the Outstation sent an Unsolicited Response	Objects appropriate to the request, if the Master sent one
Key Status	Response providing the outstation's current status of the Session Keys	0x83 Authentication Response	g120v5 Session Key Status object	0x20 Authentication Request	g120v4 Session Key Status Request object
Key Change	Changes the symmetric Session Keys subsequently used by master and outstation for authentication	0x83 Authentication Response If authentication was successful	g120v5 Session Key Status object	0x20 Authentication Request	g120v6 Session Key Change object
		0x83 Authentication Response If authentication failed	g120v7 Authentication Error object		

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

IEC/TS 62351-5 message	Description	Message from outstation contains		Message initiated because master sent	
		DNP3 function codes	DNP3 objects	DNP3 function codes	DNP3 objects
Error	Indicates authentication provided in the previous challenge reply from master was incorrect or that an aggressive mode request did not authenticate	0x83 Authentication Response If authentication failed	g120v7 Authentication Error object	0x20 Authentication Request	g120v2 Authentication Reply object
				Any valid function code	g120v3 Authentication Aggressive Mode Request object. • • • Objects appropriate for standard DNP3 request • • • g120v9 (with error) Authentication HMAC object
NOTE—Shaded cells indicate optional messages.					

7.3.1.3 DNP3 sequence numbering

Each DNP3 authentication Challenge, Reply, or Error message shall have the same DNP3 application sequence number as the critical DNP3 fragment that was challenged.

Figure 37 illustrates how this rule would be applied for a Select/Operate control sequence using challenge and reply. DNP3 application sequence numbers are shown in parentheses. Figure 38 illustrates that when the aggressive mode is used, the sequence numbering is identical to normal non-authenticated DNP3. Figure 39 illustrates an example of a failed DNP3 select/operate authentication.

Implementers should note that the additional challenge and reply messages may require that certain timing parameters, such as the select–operate timeout, be increased.

Unless otherwise described in this subclause, normal DNP3 sequence numbering rules apply.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 37—Example of DNP3 select/operate authentication

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 38—Example of DNP3 select/operate authentication in aggressive mode

Figure 39—Example of failed DNP3 select/operate authentication

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.3.1.4 More DNP3 message examples

Figure 40 illustrates several of the messages described in the preceding subclauses as part of a typical initialization sequence. The outstation generates an unsolicited response to notify the master that it has restarted. Rather than confirm the unsolicited response, the master first initializes the Session Keys. Then, when the outstation re-attempts the restart unsolicited response, the master authenticates it before supplying a confirmation. The confirmation is not authenticated, but the outstation requires authentication of the Write operation. Following this sequence, both sides are permitted to use the aggressive mode because a complete challenge-reply sequence has taken place in both directions.

Figure 40—Example of DNP3 initialization sequence

Figure 41 illustrates how a poll-response sequence could be authenticated in aggressive mode. Figure 42 illustrates an example of failed authentication of outstation data. The poll is not considered critical by the outstation, but the confirmation is. Since Responses and Confirms are optionally critical functions, it is up to the master to require that the Analog Change Events are critical, and up to the outstation to require that Confirms are critical.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 41—Example of DNP3 authentication of outstation polling data

Figure 42—Example of failed authentication of outstation data

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.3.1.5 DNP3 state machine overviews

Figure 43 and Figure 44 are reproductions of Figure 35 and Figure 36, respectively, but with DNP3 function codes and object variations substituted for the authentication message names. They are provided here to illustrate how the IEC/TS 62351-5 authentication mechanism is mapped to DNP3. Note that the top portion of the diagram remains the same for both master and outstation although the function codes used for transmitting and receiving the object variations are reversed. As noted previously, these overview diagrams show only the major transitions; the tables in 7.3.2 are definitive.

Tx = Function Code **0x20** Authentication Request
Except **g120v7** uses **0x21** Authentication Request – No Ack
Rx = Function Code **0x83** Authentication Response if it is a security message

Figure 43—Master state machine showing DNP3 function codes and object variations

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Tx = Function Code **0x83** Authentication Response
Rx = Function Code **0x20** Authentication Request if it is a security message

Figure 44—Outstation state machine showing DNP3 function codes and object variations

7.3.2 Formal procedures

This subclause formally describes the procedures used by devices implementing this authentication mechanism as a part of each protocol. If this subclause differs from 7.2, this subclause shall be considered definitive.

The state machines in this subclause describe the protocol in terms of IEC/TS 62351-5 “messages.” Refer to 7.3.1 for a description of how each of these messages is implemented using DNP3 function codes and objects, in each direction.

Table 38 describes the states used by these state machines, in the general order in which they might be expected to occur. Refer to Figure 35 and Figure 36 for an overview of how the state machines work together.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 38—States used in the state machine descriptions

State	Implemented in		Description	Refer to table
	Master	Outstation		
Wait for Key Status	YES	No	The master has either just initialized, or its Session Keys have expired. It has just transmitted a Request Key Status message and is waiting for the outstation to transmit a Key Status message.	Table 44
Wait for Key Change Confirmation	YES	No	The master has transmitted a Key Change message and is waiting for the outstation to send confirmation that the Key Change has been accepted, by transmitting a Key Status message with the Key Status = <1> OK	Table 44
Wait for Reply	YES	YES	The Session Keys have been initialized and an authentication is in progress. One of the devices has transmitted a Challenge message and is waiting for the other end to transmit a Reply message. The critical ASDU that was challenged has been queued, waiting to be processed if the Reply is valid. It shall be discarded if the Reply is invalid or an error condition occurs	Table 40
Security Idle	YES	YES	There is no authentication in progress. The device is executing the standard DNP3 protocol. The Session Keys may or may not be initialized.	Table 40

In each of these states except Security Idle, the device is waiting for a reply concerning a particular user. Devices shall keep a separate set of timers and states for each user. However, *only one user may be in a state other than Security Idle at a time.*

As illustrated in Figure 45, if an event occurs in a state other than Security Idle, and the event concerns a user other than the one that entered that state, the device shall either queue the event or treat it as an error, as described in the state machines.

As an example of queuing, consider the case of initialization. If there are three users at the master, the master begins by sending a Request Key Status message for the first user and entering *Wait for Key Status* for that user. The initialization event for the other two users is queued by the master. The second user does not enter *Wait for Key Status* until the first user has reached *Security Idle* by either succeeding—or failing—to initialize its Session Keys.

As a different example, consider the case when an outstation is in *Wait for Reply* state waiting for User 1 to reply to a challenge, when the master unexpectedly sends an Aggressive Mode Request message for User 2. Per rows 15 through 17 of Table 40, the outstation stops waiting for the master to reply regarding User 1 and processes the request for User 2. The outstation does not process messages for two users at the same time.

Any events not covered by these state machines and procedures shall be ignored.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 45—Behavior model for multiple users

If the Maximum Error Count is exceeded in any of the non-idle states, the device shall return to the Security Idle state and may process events associated with different users. Devices may not process DNP3 request or response messages associated with the user whose error count exceeded the maximum until the master re-initializes the Session Keys for that user. This requirement is important because it helps to prevent an attacker from denying service by forcing the device to process a particular user continuously.

7.3.2.1 Challenger procedures

7.3.2.1.1 Challenger role

A device, either master or outstation, that requires authentication from the other device in order to communicate shall be called a Challenger. Challengers shall issue Challenge messages in reply to Critical ASDUs, according to the state machine described in Table 40.

Challengers shall never intentionally retransmit the same Challenge message. Any time a Challenge is issued, it shall be created using new Challenge Data and a new Challenge Sequence Number.

Note that in order to reach either of the two states described in Table 40, the Challenger shall have established a set of Session Keys using the Master state machine in Table 44.

Each fragment of a multi-fragment Response shall be challenged individually. If a master issues a challenge to a multi-fragment response (function codes 129 or 130) from an outstation, it shall issue a challenge for each fragment in the response.

The CON (confirm) bit is never set in Authentication Response fragments transmitted by the outstation unless multiple fragments are sent, or an aggressive mode confirmation is requested, as described in 7.3.2.1.2.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.3.2.1.2 Critical functions

Each Challenger shall distinguish between Critical ASDUs and Non-Critical ASDUs. A Critical ASDU shall be a message implementing a function that the Challenger requires to be authenticated. IEC/TS 62351-5 states the following minimum requirements:

— Outstations shall consider all output operations (controls, setpoint adjustments, parameter settings, etc.) to be critical.

— Challengers may optionally consider additional functions beyond this minimum subset to be critical.

The following rules are used to identify the DNP3 operations that shall be considered critical, requiring authentication.

a) Any Application Layer DNP3 Requests identified with a "MANDATORY" in the "Critical" column of Table 39 shall be considered critical operations. DNP3 outstations complying with this authentication mechanism shall require masters to authenticate all DNP3 Request fragments that contain these function codes.

b) DNP3 outstations may optionally require authentication of any other Request fragments.

c) There are no DNP3 Responses that are mandatory for a DNP3 master to designate as critical operations. DNP3 masters may optionally require authentication of any DNP3 Response, but they are not required to do so for compliance.

d) If a DNP3 device claims compliance with this authentication mechanism, information identifying those function codes and objects the device considers critical, requiring authentication, shall accompany the Device Profile Document for the device.

e) Implementers of outstations should note that if an outstation considers Confirm (0x00) function codes to be critical and issues a challenge, the master may not be expecting the message. On half-duplex or multi-drop links, it is possible that the challenge would collide with the master's next request as shown in Figure 46. Outstations that wish to consider confirmations critical shall include an Authentication Challenge object (g120v1) as the last object in the response to be confirmed, as shown in Figure 47. Master stations receiving a response with the CON bit set and an Authentication Challenge object included shall issue the confirmation in aggressive mode.

   Note that this case is an exception to the rule that a complete Challenge-Reply shall be completed before aggressive mode can be used. There is sufficient information in the Challenge object to create the aggressive mode HMAC, and the aggressive mode confirmation therefore essentially constitutes a Reply.

f) Any device may arbitrarily decide that a fragment is critical and can therefore initiate a challenge for any reason. The burden is on the replying station to process the challenge regardless of whether it is expected.

g) Any messages capable of changing security configuration parameters shall be considered critical.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 46—Possible collision of confirmation challenge and next master request

Figure 47—Preventing confirmation challenge collisions using aggressive mode

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 39—DNP3 critical request function codes

Function code		Description	Critical
Decimal	Hex		
0	0x00	Confirm	optional
1	0x01	Read	optional
2	0x02	Write	MANDATORY
3	0x03	Select	MANDATORY
4	0x04	Operate	MANDATORY
5	0x05	Direct Operate	MANDATORY
6	0x06	Direct Operate—No Acknowledgment	MANDATORY
7	0x07	Immediate Freeze	optional
8	0x08	Immediate Freeze—No Acknowledgment	optional
9	0x09	Freeze-and-Clear	optional
10	0x0A	Freeze-and-Clear—No Acknowledgment	optional
11	0x0B	Freeze-at-Time	optional
12	0x0C	Freeze-at-Time—No Acknowledgment	optional
13	0x0D	Cold Restart	MANDATORY
14	0x0E	Warm Restart	MANDATORY
15	0x0F	Initialize Data (obsolete)	optional
16	0x10	Initialize Application	MANDATORY
17	0x11	Start Application	MANDATORY
18	0x12	Stop Application	MANDATORY
19	0x13	Save Configuration (deprecated)	optional
20	0x14	Enable Unsolicited Responses	MANDATORY
21	0x15	Disable Unsolicited Responses	MANDATORY
22	0x16	Assign Class	optional
23	0x17	Delay Measurement	optional
24	0x18	Record Current Time	MANDATORY
25	0x19	Open File	optional
26	0x1A	Close File	optional
27	0x1B	Delete File	optional
28	0x1C	Get File Information	optional
29	0x1D	Authenticate File	MANDATORY
30	0x1E	Abort File	optional
31	0x1F	Activate Configuration	MANDATORY
32	0x20	Authentication Request (new)	Not applicable
33	0x21	Authentication Request—No Ack (new)	Not applicable
129	0x81	Response	optional
130	0x82	Unsolicited Response	optional
131	0x83	Authentication Response (new)	Not applicable

7.3.2.1.3 Authentication procedures

If the Challenger is in the states *Wait for Key Status*, or *Wait for Key Change Confirmation*, when it receives a Reply message, it shall consider the Reply message to be an *Rx Invalid Reply* event because the Session Keys are not valid. Similarly, if the Challenger receives an Aggressive Mode Request in any of these states, the Challenger shall consider it to be an *Rx Invalid Aggressive Mode Request* event.

Upon receiving a Reply message, the Challenger shall calculate the HMAC Value from the information it transmitted in the Challenge message, as described in the definition of the Authentication Challenge object.

If the HMAC Value from the Reply matches the calculated HMAC Value, and the Challenge Sequence Numbers from the Challenge and Reply messages also match, the Challenger shall consider the Reply message to be an *Rx Valid Reply* event.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Otherwise, the Challenger shall consider the Reply message to be an *Rx* I*nvalid Reply* event.

Upon receiving an ASDU containing an Aggressive Mode Request, the Challenger shall calculate the HMAC Value from the information in the ASDU as described in the definition of the Authentication Aggressive Mode Request object. If the HMAC Value in the Aggressive Mode Request matches the calculated HMAC Value and the Challenge Sequence Number in the Aggressive Mode Request is correct as described in the definition of the Authentication Aggressive Mode Request object, the Challenger shall consider the ASDU to be an *Rx Valid Aggressive Mode Request* event.

Otherwise, the Challenger shall consider the Aggressive Mode Request message to be an *Rx Invalid Aggressive Mode Request* event.

In particular, the Challenger shall consider any Aggressive Mode Request to be an *Rx Invalid Aggressive Mode Request* event if the Challenger has not previously received at least one *Rx Valid Reply* event from the Responder. This rule follows from the definition of the Aggressive Mode Request because the Challenge Sequence Number in an Aggressive Mode Request is derived from the Challenge Sequence Number found in the Challenge most recently received by the Responder.

7.3.2.1.4 Challenger state machine

Challengers (either master or outstation) shall implement the state machine described in Table 40. Note that whenever the outstation sets the Key Status to a value other than OK, the set of Session Keys for the identified user shall be considered invalid and all authentication attempts for that user shall fail until the Key Status is OK again.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 40—Challenger state machine

Event	Event description	State				
		Security idle		Wait for reply		
		The device is executing the standard DNP3 protocol.		The device is waiting for the other device to authenticate itself.		
		Action	Next state	Action	Next state	
A	B	C	D	E	F	
Rx Unsolicited Non-Critical ASDU	MASTER ONLY. The master receives an unsolicited ASDU that does not require authentication.	Process the ASDU and issue a Confirm.	Security Idle	Process the ASDU and issue a Confirm.	Wait for Reply	1
Rx Non-Critical ASDU	The Challenger receives an ASDU that does not require authentication.	Process the Non-Critical ASDU and transmit any appropriate response required by the standard protocol.	Security Idle	Discard the ASDU that was queued pending authentication. Increment the Error count. Cancel the Reply Timer. If the Error Count is less than the Maximum Error Count , Process the non-critical ASDU and transmit any appropriate response required by the standard protocol. If the Error Count equals the Maximum Error Count, Transmit an Error Message with reason <2> Unexpected Reply. If the Maximum Error Count has been exceeded, behave according to the “Max Invalid Replies” event.	Security Idle	2
Rx Critical ASDU	The Challenger receives an ASDU that requires authentication.	If (MASTER and Session Keys are valid) OR If OUTSTATION: Increment the Challenge Sequence Number (CSQ). Create and transmit a Challenge message calculated from the Critical ASDU. Start the Reply Timer. Queue the Critical ASDU for execution later.	Wait for Reply	Discard the previous Critical ASDU pending authentication. Increment the Error Count. If the Error Count is less than the Maximum Error Count, Increment the Challenge Sequence Number (CSQ) Create and transmit a Challenge message calculated from the Critical ASDU. Start the Reply Timer. Queue the Critical ASDU for execution	Wait for Reply	3

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Event	Event description	State				
		Security idle		Wait for reply		
		The device is executing the standard DNP3 protocol.		The device is waiting for the other device to authenticate itself.		
		Action	Next state	Action	Next state	
A	B	C	D	E	F	
		If MASTER and Session Keys are Invalid: Transmit a Key Status Request Message. Start the Reply Timer.	Wait for Key Status (Table 44)	later. If the Error Count is equal to the Maximum Error Count, Transmit an Error Message with reason <2> Unexpected Reply. If the Maximum Error Count has been exceeded, behave according to the "Max Invalid Replies" event.	Wait for Reply	4
Rx Valid Reply	The Challenger receives a Reply message that correctly authenticates the other device based on the most recently transmitted Challenge.	Discard the message.	Security Idle	If a critical ASDU is queued awaiting the challenge response, then process the ASDU and transmit the ASDU's response. Cancel the Reply Timer. Reset the Error Count to zero.	Security Idle	5
Rx Invalid Reply	The Challenger receives a Reply message that does not correctly authenticate the other device.	Discard the message.	Security Idle	Increment the Error count. Discard the ASDU that was queued pending authentication. If the Maximum Error Count has not been exceeded, transmit an Error Message with reason <1> Authentication Failed. Cancel the Reply Timer. If the Maximum Error Count has been exceeded, behave according to the "Max Invalid Replies" event below.	Security Idle	6

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Event	Event description	State				
		Security idle		Wait for reply		
		The device is executing the standard DNP3 protocol.		The device is waiting for the other device to authenticate itself.		
		Action	Next state	Action	Next state	
A	B	C	D	E	F	
Reply Timeout	The Reply Timer started when entering Wait for Reply state has expired. This may be the standard response timer for the protocol. Refer to clause 7.4.1.4.1 for details regarding the Reply Timer.	Should not occur—this is an error condition.	Security Idle	Increment the Error count. Cancel the Reply Timer. Discard any ASDUs that were queued pending an authentication Reply. If the Maximum Error Count has not been exceeded, transmit an Error Message with reason <3> No Reply. If the Maximum Error Count has been exceeded, behave according to the "Max Invalid Replies" event below.	Security Idle	7
Max Invalid Replies Or Comm Failure Detected	The Maximum Error Count has been exceeded. This event affects a single user. The protocol has detected a communications failure for some other reason. This event affects all users. Refer to 7.4.1.4.1 for details regarding the Reply Timer.	MASTER: Transmit a Key Status Request Message. Start the Reply Timer.	Wait for Key Status (Table 44)	MASTER: Discard the critical ASDU that was queued pending authentication. Transmit a Key Status Request Message. Start the Reply Timer.	Wait for Key Status (Table 44)	8
		OUTSTATION: Set the current Key Status to COMM_FAIL. If communications failure occurred, or NOT INIT otherwise.	Security Idle	OUTSTATION: Discard the critical ASDU that was queued pending authentication. Set the current Key Status to AUTH_FAIL	Security Idle	9
Rx Error Message	The Challenger receives an Error Message.	Log the Error message, noting that the message was unexpected.	Security Idle	Log the error message. If the Error Code is <5> HMAC algorithm Not Permitted, use a different HMAC algorithm to send the next Challenge. Do not send another Challenge immediately, but wait for an appropriate event to cause the Challenge. Discard the pending ASDU and cancel the Reply Timer.	Security Idle	10
Key Change Timeout	For MASTER only. Either the Key Change timer has expired, or The Key Change Count has been exceeded. Refer to 7.4.1.4.	Transmit a Key Status Request Message. Start the Reply Timer.	Wait for Key Status (Table 44)	Queue the event and process it after returning to Security Idle state.	Wait for Reply	11

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Event	Event description	State				
		Security idle		Wait for reply		
		The device is executing the standard DNP3 protocol.		The device is waiting for the other device to authenticate itself.		
		Action	Next state	Action	Next state	
A	B	C	D	E	F	
Expected Key Change Timeout	For OUTSTATION only. The outstation has not received a valid Key Change message within the Session Key Change interval or Session Key Change count configured at the outstation. Refer to 7.4.1.4.	Set Key Status = NOT_INIT Invalidate current session keys.	Security Idle	IF the timeout is for the waiting user, Set Key Status = NOT_INIT Invalidate current session keys Discard the pending ASDU. Cancel the Reply Timer.	Security Idle	12
				IF the timeout is not for the waiting user, Set Key Status for the user specified in the timeout to NOT_INIT	Wait for Reply	13
Rx Key Status Request	For OUTSTATION only. The outstation receives a Key Status Request message.	Transmit a Key Status message containing the current Key Status.	Security Idle	Discard the critical ASDU pending authentication. Transmit a Key Status message Containing the current Key Status. Cancel the Reply Timer.	Security Idle	14
Rx Valid Aggressive Mode Request	The Challenger receives an ASDU containing an Aggressive Mode Request message that correctly authenticates the other device.	IF Aggressive Mode is enabled: Perform the operations specified in the ASDU containing the Aggressive Mode Request and transmit any appropriate response required by the standard protocol.	Security Idle	IF Aggressive Mode is enabled: Discard the previously pending Critical ASDU. Perform the operations specified in the ASDU containing the Aggressive Mode Request and transmit any appropriate response required by the standard protocol. Cancel the Reply Timer.	Security Idle	15
		IF Aggressive Mode is disabled, discard the request and Increment the Error Count. If the Maximum Error Count has not been exceeded, transmit an Error Message with reason <4> Aggressive Mode Not Supported.	Security Idle	If Aggressive mode is disabled, Discard the critical ASDU that was queued pending authentication and discard the Aggressive Mode Request. Increment the Error Count. If the Maximum Error Count has not been exceeded, transmit an Error Message with reason <4> Aggressive Mode Not Supported. Cancel the Reply Timer	Security Idle	16

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Event	Event description	State				
		Security idle		Wait for reply		
		The device is executing the standard DNP3 protocol.		The device is waiting for the other device to authenticate itself.		
		Action	Next state	Action	Next state	
A	B	C	D	E	F	
Rx Invalid Aggressive Mode Request	The Challenger receives an ASDU containing an Aggressive Mode Request that does not correctly authenticate the other device. Note that all Aggressive Mode Requests are invalid until at least one valid challenge reply has been received.	Increment the Error Count. If the Maximum Error Count has not been exceeded, transmit an Error Message with reason <1> Authentication Failed if Aggressive Mode is enabled, or with reason <4> Aggressive Mode Not Supported otherwise.	Security Idle	Discard the previously pending Critical ASDU and discard the Aggressive Mode Request. Increment the Error Count. If the Maximum Error Count has not been exceeded, transmit an Error Message with reason <2> Unexpected Reply if Aggressive Mode is enabled, or with reason <4> Aggressive Mode Not Supported otherwise. Cancel the Reply Timer.	Security Idle	17
Rx Valid Key Change	For OUTSTATION only. The outstation receives a correctly authenticated Key Change message.	Store new keys. Set Key Status = OK. Transmit Key Status message. Reset Error Count to zero.	Security Idle	Discard the previously pending Critical ASDU. Store new keys. Set Key Status = OK. Transmit Key Status message. Cancel the Reply Timer. Reset Error Count to zero.	Security Idle	18
Rx Invalid Key Change	For OUTSTATION only. The outstation receives an improperly authenticated Key Change message.	Set Key Status = AUTH_FAIL. Transmit Key Status message.	Security Idle	Discard the invalid Key Change message.	Wait for Reply	19
Rx Challenge	The device receives a Challenge message.	Reply as described in 7.3.2.2.	Security Idle	Increment the Error count. Cancel the Reply Timer. If the Maximum Error Count has not been exceeded, transmit an Error Message with reason <2> Unexpected Reply. Cancel the Reply Timer. If the Maximum Error Count has been exceeded, behave according to the “Max Invalid Replies” event.	Security Idle	20

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.3.2.1.5 Error messages

As described more formally in Table 40, devices may initially respond to error conditions by transmitting error messages. To help protect against denial-of-service attacks, a device shall stop transmitting error messages after it has counted a number of errors that exceeds a preset Maximum Error Count, described in 7.4.1.4.6. Devices may also choose to not send error messages at any time, regardless of error count.

Note that error messages may be transmitted on a DNP3 association other than the one on which authentication is performed. This may be extremely useful for detecting attacks and is therefore recommended. It is also recommended that all errors be logged.

Table 41 illustrates how error messages are implemented in DNP3.

Table 41 —Use of error message objects in DNP3

Function code	Object variation	Behavior	Purpose
0x83 Authentication Response 0x21 Authentication Request—No Ack	g120v7 Authentication Error (as an Info object)	Response to a security message on the same DNP3 association	Notify other device of a possible configuration error or lack of synchronization
0x81 Response 0x82 Unsolicited Response	g120v7 Authentication Error (as an Event object)	Included with other DNP3 data in an event response on the other DNP3 association	Detection of an attack (see NOTE)
NOTE—When the error message is sent as an event using function codes 0x81 or 0x82, the CON bit shall be set. The rule regarding CON bits in Authentication Responses in 7.3.2.1.1 does not apply.			

7.3.2.2 Responder procedures

7.3.2.2.1 Responder role

A device, either master or outstation, that supplies authentication data shall be called a Responder. Each Responder shall follow the procedures described in this subclause.

7.3.2.2.2 Responding to challenges

A Responder shall respond to a Challenge message with a correctly formed Reply message within an acceptable Reply Timeout defined per system as described in 7.4.1.4.1.

If the following conditions are met:

— The Responder is a master.

— The Authentication Challenge object is the last object in the response, but the response also contains other data to be confirmed.

— The CON bit is set in the response.

Then, the master shall, instead of transmitting an Authentication Reply object, transmit a Confirm (0x00) message in aggressive mode, i.e., containing:

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— Function Code 0x00 Confirm

— Authentication Aggressive Mode Request object (g120v3)

— Authentication HMAC object (g120v9)

The rationale for this rule and an example are found in 7.3.2.1.2.

A Responder shall not proceed with further communications until it has successfully responded to the Challenge message. This rule includes not responding to any subsequent Challenge messages until the current Challenge is completed.

7.3.2.2.3 Aggressive mode

Aggressive mode, in which a device supplies authentication information in the same ASDU as the data it is authenticating, shall be implemented by all DNP3 devices conforming to this specification.

These devices shall also provide a mode of operation in which aggressive mode is disabled.

A Responder that uses aggressive mode shall place a correctly formed Aggressive Mode Request within the ASDU being authenticated. The location and the formatting of the Aggressive Mode Request within the ASDU shall be specific to the protocol and described in the specifications of the affected protocol.

A Responder shall not transmit an Aggressive Mode Request until it has successfully responded to at least one Challenge message.

7.3.2.3 Master procedures

7.3.2.3.1 Master role

In addition to acting as a Challenger and a Responder, masters shall follow the procedures described in this subclause in order to initialize and change keys at the outstation.

7.3.2.3.2 Changing session keys

There shall be two Session Keys, one used for authenticating data in the Monitoring Direction and one for authenticating data transmitted in the Control Direction, as described in Table 35. The outstation and masters shall maintain a unique set of Session Keys for each user of the outstation, and a default set of Session Keys shall be used for cases when the master acts for multiple users (as in the case of a poll, for instance), or when the outstation initiates the security message sequence.

Each master shall initialize the Session Keys upon establishing communications and periodically change the Session Keys as described in Table 44. The change interval shall be set using a configurable parameter as discussed in 7.4.1.4.2.

The master shall use a symmetric Update Key to encrypt the Session Keys and transmit them to the outstation in a Key Change message. There shall be a separate Update Key for each user of the outstation and a default Update Key for the default set of Session Keys.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.3.2.3.3 Assigning user numbers

The master and outstation shall identify with a unique User Number for each set of Session Keys that they share. Each security message contains the applicable User Number and therefore specifies the applicable set of Session Keys. The purpose of User Numbers is to make individual human beings accountable for the critical operations they perform remotely on the outstation.

Figure 48 shows an example of how User Numbers may be assigned. In this example, there are two masters, each of which has two users, communicating with a single outstation. When a user initiates a critical operation that sends DNP3 messages from a master to an outstation, the master is responsible for authenticating that operation using the appropriate User Number and corresponding Session Keys for that user.

Figure 48—Example of user number and association ID assignments

Note that in the example, both Bob and Donald have the same User Number. User Numbers need not be unique within the entire outstation, only within a particular DNP3 association. The precise definition of a DNP3 association is found in Clause 13, but it essentially means the logical connection between a particular master and a particular outstation.

In the example, there are two associations, one from the single outstation to each master. The figure shows a separate software process for each association, but this is done purely for illustration; the associations may be implemented many different ways. Each association uses a particular set of addresses for the master and the outstation. The addresses may include DNP3 Data Link Layer addresses, IP addresses, TCP or UDP port numbers, serial port numbers, and internal software identifiers.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Within a particular association, the User Number uniquely identifies a particular user. For most of the security messages, this is sufficient identification. However, there is one exception. As noted in 7.3.2.1.5, this specification recommends that error messages be transmitted on associations other than that on which the error occurred, in order to help with intrusion detection.

Devices shall therefore include in each error message an Association ID, which is an integer uniquely identifying the DNP3 association and the complete set of addresses it uses. The combination of Association ID and User Number shall uniquely identify a user with an entire device. Table 42 shows that in the example, the Association IDs are simply the numbers "1" and "2."

Table 42—Example of user number and association ID assignments

Association ID	User number	User	Master	Outstation process
1	0	Unknown	X	A
1	1	Default	X	A
1	7	Alice	X	A
1	12	Bob	X	A
2	0	Unknown	Y	B
2	1	Default	Y	B
2	5	Charlie	Y	B
2	12	Donald	Y	B

Note that in each association, the outstation uses the reserved User Number <0> when it is issuing a Challenge but does not yet know the user associated with the critical ASDU it is challenging. The outstation and master also use a second reserved User Number <1> as the default User Number in a number of different situations, as illustrated in Table 43.

Table 43—When to use the reserved user numbers

Case	User number
Outstation sends Challenge	Unknown <0>
Outstation sends Aggressive Mode unsolicited response	Default <1>
Outstation sends Aggressive Mode response	Default <1>
Master Challenges unsolicited response from outstation	Default <1>
Master Challenges response from outstation	Default <1>
Master sends common request (e.g., poll) for data to be processed by multiple users.	Default <1>
Any other case	<2…65535>

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.3.2.3.4 Master state machine

The master shall execute the state machine in Table 44.

Table 44—Master state machine

Event	Event description	State				
		Wait for key status		Wait for key change confirmation		
		The master is waiting for the outstation to send any Key Status message.		The master is waiting for the outstation to send confirmation that the Key Change has been accepted, by transmitting a Key Status message with the Key Status = <1> OK		
		Action	Next state	Action	Next state	
A	B	C	D	E	F	
Init	The master has initialized	Transmit a Key Status Request message. Start the Reply Timer.	Wait for Key Status	Not possible.	Not possible.	1
Rx Key Status <> OK	The master receives a Key Status message with the Key Status set to a value other than <1> OK.	Transmit a Key Change message. Start the Reply Timer.	Wait for Key Change Confirmation	Transmit a Key Status Request message. Start the Reply Timer.	Wait for Key Status.	2
Rx Key Status = OK	The master receives a Key Status message with the Key Status set to <1> OK.	Transmit a Key Change message. Start the Reply Timer.	Wait for Key Change Confirmation	Start the Key Change timer and reset the Key Change counter. Reset the Error Count.	Security Idle	3
Reply Timeout	The Reply Timer has expired while the master was waiting for a response from the outstation.	Transmit a Key Status Request message. Start the Reply Timer.	Wait for Key Status	Transmit a Key Status Request message. Start the Reply Timer.	Wait for Key Status	4
Key Change Timeout	Either the Key Change Timer has expired on the master, or the number of transmitted or received protocol messages has exceeded the Message Count Threshold. This event should not happen in either of these states for the current user.	IF the timer is for the same user, discard. IF the timer is for a different user, queue the event and process it when next in Security Idle state.	Wait for Key Status	IF the timer is for the same user, discard. IF the timer is for a different user, queue the event and process it when next in Security Idle state.	Wait for Key Change Confirmation	5

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Event	Event description	State				
		Wait for key status		Wait for key change confirmation		
		The master is waiting for the outstation to send any Key Status message.		The master is waiting for the outstation to send confirmation that the Key Change has been accepted, by transmitting a Key Status message with the Key Status = <1> OK		
		Action	Next state	Action	Next state	
A	B	C	D	E	F	
Rx Challenge message	The master receives a Challenge message or Aggressive Mode Request message even though the Session Keys are not yet valid.	Increment the Error Count. If the Maximum Error Count has not been exceeded, transmit an Error Message with reason <1> Authentication Failed	Wait for Key Status	Increment the Error Count. If the Maximum Error Count has not been exceeded, transmit an Error Message with reason <1> Authentication failed	Wait for Key Change Confirmation	6
Rx Critical ASDU	The master receives an ASDU that requires authentication even though the Session Keys are not yet valid.	Increment the Error Count. If the Maximum Error Count has not been exceeded, transmit an Error Message with reason <1> Authentication Failed	Wait for Key Status	Increment the Error Count. If the Maximum Error Count has not been exceeded, transmit an Error Message with reason <1> Authentication failed	Wait for Key Change Confirmation	7
Rx Unsolicited Non-Critical ASDU	The master receives an ASDU that does not require authentication and was spontaneously transmitted by the outstation.	Process the ASDU and issue a Confirm if required.	Wait for Key Status	Process the ASDU and issue a Confirm if required.	Wait for Key Change Confirmation	8
Rx Inappropriate Non-Critical ASDU	The master receives a non-critical, non-authentication ASDU in response to its previous authentication message, or receives some other indication that the outstation may not be capable of processing authentication messages.	Process the ASDU and issue a Confirm if required. Start the Key Change Timer and reset the Key Change Counter.	Security Idle	Process the ASDU and issue a Confirm if required. Start the Key Change Timer and reset the Key Change Counter.	Security Idle	9
A User Wants to Transmit an ASDU	A user wishes to transmit from this master. May be either a critical or non-critical ASDU.	Queue the ASDU until the next time the master enters Security Idle state.	Wait for Key Status	Queue the ASDU until the next time the master enters Security Idle state.	Wait for Key Change Confirmation	10

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Event	Event description	State				
		Wait for key status		Wait for key change confirmation		
		The master is waiting for the outstation to send any Key Status message.		The master is waiting for the outstation to send confirmation that the Key Change has been accepted, by transmitting a Key Status message with the Key Status = <1> OK		
		Action	Next state	Action	Next state	
A	B	C	D	E	F	
Rx Invalid Key Status	The master receives a Key Status message for a user other than the one which is currently in Wait for Key Status or Wait for Key Change Confirmation state. This event should not occur because the outstation should be responding to a request for THIS user. OR Receives a Key Status = OK, but the message is not authentic.	Increment the Error Count. If the Maximum Error Count has not been exceeded, transmit an Error Message for the original user with reason <1> Authentication Failed; Transmit a Key Status Request message for the original user. Start the Reply Timer.	Wait for Key Status	Increment the Error Count. If the Maximum Error Count has not been exceeded, transmit an Error Message for the original user with reason <1> Authentication Failed; Transmit a Key Status Request message for the original user. Start the Reply Timer.	Wait for Key Status	11
Maximum Error Count Exceeded	As a result any of the other events, the Maximum Error Count for this user was exceeded. The master has unsuccessfully tried several times to reset the Session Keys for this user. Shall give another user a chance to initialize keys.	Start the Key Change Timer. Reset the Key Change Counter.	Security Idle	Start the Key Change Timer. Reset the Key Change Counter.	Security Idle	12

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.3.2.4 Outstation procedures

7.3.2.4.1 Outstation role

In addition to acting as a Challenger and a Responder, each outstation shall follow the procedures described in this subclause, permitting the Master to initialize and change Session Keys.

7.3.2.4.2 Key status

The outstation shall maintain an internal variable having the possible values of Key Status described in the definition of the Authentication Key Status object, and return this value in response to each Key Status Request Message.

7.3.2.4.3 Authenticating session key changes

Upon receiving a Key Change message, the outstation shall unwrap the Key Wrap Data in the Key Change message using the current Update Key.

If the Key Status information in the Key Wrap Data matches the last Key Status information transmitted by the outstation, the outstation shall consider the Key Change message authentic and valid.

If any of the unwrapped Key Status information does not match the last Key Status information transmitted by the outstation, the outstation shall consider the Key Change message invalid.

7.3.2.4.4 Changing session keys

An outstation shall respond to a Key Change message within an acceptable Reply Timeout defined per system as described in 7.4.1.4.1.

An outstation shall be configured with a timer such that it shall invalidate a set of session keys if it has not received a Key Change message within that interval as described in 7.4.1.4.4

7.4 Interoperability requirements

This subclause describes which capabilities shall be considered to fall into the following categories:

— Mandatory to verify interoperability between devices

— Optional but required to be tested if implemented

— Recommended practices

7.4.1 Minimum requirements

This subclause describes the mandatory minimum capabilities required for a device to comply with this specification.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.4.1.1 HMAC algorithms

Each device shall implement the HMAC algorithms listed in this subclause.

7.4.1.1.1 HMAC-SHA1

Each device shall permit the use of HMAC-SHA1, as described in 7.1.2 to calculate the HMAC Value. The HMAC Value shall be the 160 bits (20 octets) output of the HMAC algorithm, truncated to either the leftmost 4 octets or the leftmost 10 octets. If this specification is used over TCP/IP, the truncated value shall be 10 octets. If it is used over serial links, the truncated value can be 4 octets. However, the longest practical HMAC should be used whenever possible.

This shall be the default HMAC algorithm.

7.4.1.2 Key wrap algorithms

Each device shall implement key wrap algorithms as described in this subclause.

7.4.1.2.1 AES-128 key wrap

Each device shall permit the use of the Advanced Encryption Standard Key Wrap mechanism, as described in 7.1.2, to encrypt and decrypt Session Keys. The Key Encryption Key (KEK) referred to in the Key Wrap specification shall be the Update Key. The default initialization vector shall be used.

7.4.1.3 Fixed values

Each device shall fix the following parameters to comply with this specification:

7.4.1.3.1 Minimum session key size

The minimum size of the Session Keys used to calculate the HMAC Value shall be 128 bits.

7.4.1.3.2 Minimum update key size

The minimum size of the Update Key used to encrypt and decrypt Session Keys shall be 128 bits.

7.4.1.4 Configurable values

Each device shall permit changes to the parameters described in this subclause. Changes to these parameters shall be permanently retained over restarts of the device.

7.4.1.4.1 Reply timeout

The reply timeout used by devices to detect communication failures shall be settable in hundreds of milliseconds. The default value shall be 2 seconds. The maximum value shall be no less than 120 seconds.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.4.1.4.2 Session Key change interval

The Session Key change timeout used by the master to determine when to change Session Keys shall be settable in seconds up to 2 hours. The default value shall be 15 minutes. To accommodate systems that communicate infrequently (for instance, every few hours or days), it shall be possible to disable the Session Key change interval and use only the Session Key change count. Alternatively, the device may permit Session Key change intervals measured up to 1 week in length.

WARNING

Implementers should not increase the Session Key change interval beyond 30 minutes unless they also increase the size of the HMAC Value. FIPS 198 requires HMAC output to be truncated to no less than half its standard size "unless an application or protocol makes numerous trials impractical." In the case of this authentication mechanism, the requirement for frequent changes of Session Keys fulfils this criterion and makes it possible to use shorter HMAC Values.

7.4.1.4.3 Session Key change count

The master shall also change Session Keys whenever a configured number of authentication ASDUs has been transmitted in either direction since the last key change. The value shall be settable from one in increments of one. The default value shall be 1000. The maximum value shall be no less than 10 000.

7.4.1.4.4 Expected Session Key change interval and count

The outstation shall maintain a timer and a count between successive Key Change messages in the same manner as the master. The outstation shall invalidate the current set of Session Keys if they have not been changed within the configured interval. This rule shall cause the Session Keys to become invalid whenever either the master or the outstation times out, whichever happens sooner. To avoid excessive message exchanges, it is recommended that the outstation interval and count be configured for twice the interval and that the count be configured at the master.

7.4.1.4.5 Use of aggressive mode

Aggressive mode is considered optional by IEC/TS 62351-5. However, it is not optional for DNP3. All DNP3 implementations claiming conformance to this specification shall implement it. They shall also permit it to be configured as disabled. If an outstation requests aggressive mode authentication of a Confirm (0x00) message as described in 7.3.2.1.2, the master shall do so regardless of whether aggressive mode is disabled.

7.4.1.4.6 Maximum error count

Configuring the maximum error count is considered optional by IEC/TS 62351-5. However, it is not optional for DNP3. The maximum error count shall be configurable up to a maximum of 10 errors or down to 0. The default shall be 2.

7.4.1.4.7 Disabling authentication

Each device that supports this authentication mechanism shall permit this mechanism to be completely disabled by configuration on a per-association basis. It shall not be possible to change this configuration parameter remotely. The authentication mechanism shall be enabled by default.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.4.2 Options

This subclause describes capabilities that are not required for compliance with this specification but may be implemented as options. If a device implements these capabilities, they shall be verified for compliance. If a device implements any capabilities not listed in this subclause, the device shall provide a mode in which these capabilities are disabled.

7.4.2.1 HMAC algorithms

Each device may implement additional HMAC algorithms in addition to those listed as mandatory. If the device receives an error message with the Error Code <5> HMAC algorithm Not Permitted, it shall change to use a mandatory HMAC algorithm in its next Challenge. A device may be configured to reject particular mandatory HMAC algorithms, but it shall also support configuration to support all the mandatory HMAC algorithms.

7.4.2.1.1 HMAC-SHA256

It is recommended that devices permit the use of HMAC-SHA256, as described in 7.1.2, to calculate the HMAC Value. The HMAC Value shall be the 256 bits (32 octets) output of the HMAC algorithm, truncated to either the leftmost 8 octets or the leftmost 16 octets. When this authentication mechanism is used over TCP/IP, the truncated value shall be 16 octets. On serial implementations, it can be 8 octets. However, the longest practical HMAC should be used whenever possible.

7.4.2.2 Key Wrap algorithms

Each device may implement additional Key Wrap algorithms in addition to those listed as mandatory. If an outstation receives an error message with the Error Code <6> Encryption Algorithm Not Permitted, it shall change to use a mandatory Encryption Algorithm in its next Key Status Message.

A device may be configured to reject particular mandatory encryption algorithms, but it shall also support configuration to support all the mandatory encryption algorithms.

7.5 Special applications

This subclause defines how this specification shall be applied in a few specific situations.

7.5.1 Use with the Internet protocol suite

DNP3 implementations over TCP/IP requiring confidentiality shall use both this Application Layer mechanism and the Transport Layer Security (TLS) Internet standard as described in 7.6.

DNP3 implementations over UDP/IP shall use this Application Layer authentication mechanism by itself. Some implementations may choose to use other security measures (such as IPSec) along with this specification, but they are out of the scope of this standard.

Figure 49 illustrates the standard protocol profiles that are permitted using Secure Authentication. Shaded areas indicate security measures. Implementations that support the Confidential TCP profile shall also support the Authenticated TCP profile.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 49—Valid profiles using the secure authentication mechanism

When operating over IP, it may be possible to change and distribute Update Keys by making use of other IP-based security protocols. The DNP User's Group intends to develop specifications for this purpose. However, such mechanisms are also outside the scope of this document as of the date of its publication.

7.5.2 Use with redundant channels

When used with redundant channels, the communications shall not be considered to have failed and the Session Keys invalidated until all channels have been tried.

7.5.3 Use with external link encryptors

This authentication mechanism may be used along with external link encryptors to provide protection against the threat of eavesdropping.

It is recommended for simplicity that a common set of keys be used for both authentication and encryption. However, the definition of such rules is outside of the scope of this standard.

7.5.4 Use with data concentrators

This subclause describes special requirements when the authentication mechanism is used by a data concentrator.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.5.4.1 Definition of a data concentrator

Figure 50 illustrates an example of a typical system making use of a DNP3 data concentrator. A data concentrator does not pass DNP3 messages through itself intact, as a router or bridge does. Instead, a data concentrator terminates multiple connections of DNP3 or other protocols and stores the data from each direction in an internal runtime database. This permits the concentrator to filter the data, process it within internal software applications, or convert it into other protocols.

Data may pass through a data concentrator either "upstream," toward DNP3 masters, or "downstream," toward DNP3 outstations, also known as IEDs. DNP3 point numbers and DNP3 addresses used on any given connection may be the same or different than those used on any other connection. The data concentrator maps point numbers used on one connection to the corresponding point numbers on the other connections through the runtime database.

Figure 50—Example of user number assignments in a data concentrator

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.5.4.2 Authentication procedures for data concentrators

When a data concentrator makes use of DNP3 secure authentication, it shall follow the general rule that whenever it can distinguish the particular user who initiated an operation, it shall identify that user to downstream outstations. This rule is intended to deter repudiation by permitting concentrators or outstations to log which user initiated critical DNP3 operations. The following rules shall apply as clarifications of that rule:

a) The data concentrator shall identify each user having a DNP3 User Number (USR) on the upstream side with a separate DNP3 User Number on the downstream side. For instance, in Figure 50, Alice and Bob have User Numbers on both the upstream DNP3 link used by process "A" and the downstream DNP3 links used by process "C."

b) If an upstream protocol cannot distinguish individual users, the data concentrator shall identify each upstream protocol connection with a separate User Number on the downstream side. For instance, process "B" cannot distinguish whether protocol commands are initiated by Charlie or Donald. Therefore, the data concentrator uses a single User Number on the downstream DNP3 link used by process "C" to represent all users on Master 2.

c) The data concentrator shall identify local users and applications with separate User Numbers on the downstream side. For instance, local application "E" has its own User Number on the downstream DNP3 link. If the data concentrator had a local user interface capable of distinguishing different local users, it would identify each of these local users with a different User Number on the downstream DNP3 link.

d) User Numbers shall be unique within a DNP3 association. Refer to 7.3.2.3.3 for more details regarding this rule. User Numbers need not be sequential, but the combination of Association ID and User Number shall uniquely identify a user within a device, and a unique Update Key and a set of Session Keys shall be associated with that user.

e) The User Number used to identify a user on any given association may be different than that used on any other association. For instance, Alice's User Number on the upstream DNP3 link is 7, but her User Number on the link between the data concentrator and Outstation 3 is 22. She may have a different User Number on the link with Outstation 1.

f) If no points are mapped between an upstream or local user and a downstream outstation, the data concentrator need not map User Numbers either. Figure 50 illustrates that all the users identified in the diagram have the potential to make use of Outstation 3. However, not all users may have the potential to access Outstation 1, for instance, and therefore Outstation 1 may not have any User Numbers to distinguish them.

g) When upstream or local users initiate critical DNP3 requests that are passed through to downstream outstations, the data concentrator shall correctly identify the user making the request. If Alice initiates a binary output operation on a point that is mapped to Outstation 3, the data concentrator first authenticates the request with Master 1 using Alice's upstream User Number (7). Next, the data concentrator issues the corresponding binary output operation to Outstation 3, and it uses Alice's downstream User Number (22) when Outstation 3 requests the data concentrator to authenticate that request. Similarly, if the local automation process initiates a Freeze and Clear on several counters and Outstation 3 considers it a critical operation, the data concentrator shall authenticate the request with Outstation 3 specifying User Number 112.

h) When the data concentrator spontaneously or periodically initiates a critical DNP3 request on behalf of multiple users, it shall identify itself as the user using the "default" User Number=1. For instance, process "C" periodically initiates regular Class Data polls to gather data that shall later be distributed to Alice, Bob, and Charlie, even though none of those users specifically initiated the poll request. It is not mandatory that Class Data polls be considered critical. However, if Outstation 3 chose to challenge a Class Data poll from the data concentrator, the data concentrator would authenticate the poll identifying itself (User Number 1) as the initiating user.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.6 Compliance with IEC/TS 62351-3

DNP3 implementations that use TLS shall comply with the following requirements taken from IEC/TS 62351-3. Italicized text is a direct quotation from Edition 1 of that specification.[27]

7.6.1 Deprecation of non-encrypting cipher suites

Any cipher suite that specifies NULL for encryption shall not be used.

The list of deprecated suites includes, but is not limited to:

TLS_NULL_WITH_NULL_NULL

TLS_RSA_NULL_WITH_NULL_MD5

TLS_RSA_NULL_WITH_NULL_SHA

7.6.2 Mandatory cipher suite

DNP3 implementations that use TLS shall support the following cipher suite at a minimum:

TLS_DH_DSS_WITH_AES_128_SHA

7.6.3 Recommended cipher suites

It is recommended that DNP3 implementations using TLS support the cipher suites provided in Table 45. Implementations may also choose to implement cipher suites not listed here.

Table 45—Recommended cipher suite combinations

Key exchange		Encryption	Hash
Algorithm	**Signature**		
TLS_RSA_		WITH_RC4_128_	SHA
TLS_RSA_		WITH_3DES_EDE_CBC_	SHA
TLS_DH_	DSS_	WITH_3DES_EDE_CBC_	SHA
TLS_DH_	RSA_	WITH_3DES_EDE_CBC_	SHA
TLS_DHE_	DSS_	WITH_3DES_EDE_CBC_	SHA
TLS_DHE_	RSA_	WITH_3DES_EDE_CBC_	SHA
TLS_DH_	DSS_	WITH_AES_128_	SHA
TLS_DH_	DSS_	WITH_AES_256_	SHA
TLS_DH_		WITH_AES_128_	SHA
TLS_DH_		WITH_AES_256_	SHA

7.6.4 Negotiation of versions

Only TLS 1.0 corresponding to SSL version 3.1 (or higher) shall be allowable. Proposal of version prior to SSL 3.1 shall result in no connection being established.

[27] IEC/TS 62351-3 ed.1.0 "Copyright © 2007 IEC Geneva, Switzerland.www.iec.ch".

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.6.5 Cipher renegotiation

Implementations claiming conformance to this standard shall specify that the symmetric keys shall be renegotiated based upon a time period and a maximum allowed number of packets/bytes sent. It is a PIXIT issue, of the referencing standard, to specify the constraints on the renegotiation.

The renegotiation values shall be configurable.

DNP implementations using TLS shall renegotiate the TLS symmetric keys when the application layer Session Key Change Interval expires or the Session Key Change Count is exceeded. It is recommended that TLS renegotiation take place before the application layer key change.

The initiation of the change cipher sequence shall be the responsibility of the TCP entity that receives the TCP-OPEN indication (e.g. the called entity). A request to change the cipher, issued from the calling entity (e.g. the node that issued the TCP-OPEN) shall be ignored,

There shall be a timeout associated with the response to a change cipher request. A timeout of the change cipher request shall result in the connection being terminated. The timeout value shall be configurable.

DNP implementations using TLS shall use a change cipher request timeout configurable in the same range as the application security reply timeout described in 7.4.1.4.1.

7.6.6 Message authentication code

The Message Authentication Code shall be used.

NOTE—TLS has this capability specified as an option. This standard mandates the use of this capability to aid in countering and detection of man-in-the-middle attacks.

7.6.7 Certificate support

DNP3 Implementations using TLS shall comply with the following requirements for certificate management taken from IEC/TS 62351-3.

7.6.7.1 Multiple certificate authorities (CAs)

An implementation, claiming conformance to this standard, shall support more than one Certificate Authority.

DNP Implementations using TLS shall support at least four Certificate authorities.

The actual number shall be declared in the implementation's Device Profile Document.

The criteria and selection of a CA is out-of-scope of this standard.

7.6.7.2 Certificate size

A protocol, specifying the use of this standard, shall specify the maximum size of certificate allowed to be used. It is recommended that this size shall be less than or equal to 8192 bytes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 implementations using TLS shall support a minimum–maximum certificate size of 8192 octets. It is a local issue if larger certificates are supported.

An implementation that receives a certificate larger than the size that it can support shall terminate the connection.

7.6.7.3 Certificate exchange

The certificate exchange, and validation, shall be bi-directional. If either entity does not provide its certificate, the connection shall be terminated.

7.6.7.4 Certificate comparison

Certificates shall be validated by both the calling and called nodes. There are two mechanisms that shall be configurable for certificate verification.

— Acceptance of any certificate from an authorized CA

— Acceptance of individual certificates from an authorized CA

7.6.7.4.1 Verification based on CA

An implementation, claiming conformance to this standard, shall be capable of being configured to accept certificates from one or more Certificate Authorities without the configuration of individual certificates.

7.6.7.4.2 Verification based on individual certificates

An implementation, claiming conformance to this standard, shall be capable of being configured to accept specific individual certificates from one or more authorized Certificate Authorities (e.g. configured).

7.6.7.4.3 Certificate revocation

Certificate revocation shall be performed as specified in RFC 3280.

The management of the Certificate Revocation List (CRL) is a local implementation issue.

An implementation, claiming conformance to this standard, shall be capable of checking the local CRL at a configurable interval. The process of checking the CRL shall not cause an established connection to be terminated. An inability to access the CRL shall not cause the connection to be terminated.

Revoked certificates shall not be used in the establishment of a connection. An entity receiving a revoked certificate during connection establishment shall refuse the connection.

The revocation of a certificate shall terminate any connection established using that certificate.

Other standards, referencing this standard, shall specify recommended default evaluation intervals. The referencing standard shall determine the action that shall be taken if a certificate, currently in use, has been revoked.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 Implementations using TLS shall evaluate CRLs every twelve hours by default. The evaluation interval shall be configurable with hourly resolution. DNP3 devices shall terminate a connection when one of the certificates used to establish the connection is revoked.

NOTE—Through the normal application/distribution of CRL(s) connections may be terminated creating an inability to perform communications. Thus system administrators should develop certificate management procedures to mitigate such an occurrence.

7.6.7.4.4 Expired certificates

The expiration of a certificate shall not cause connections to be terminated.

An expired certificate shall not be used or accepted during connection establishment.

7.6.7.4.5 Signing

Signing through the use of RSA or DSS algorithms shall be supported. Other algorithms may be specified in standards that reference this document.

7.6.7.4.6 Key exchange

The key exchange algorithms shall support a maximum size of at least 1024 bits for the key.

Both RSA and Diffie-Hellman mechanisms shall be supported.

7.6.8 Co-existence with non-secure protocol traffic

Referencing standards shall provide a separate TCP/IP port through which to exchange TLS secured traffic. This shall allow for the possibility of un-ambiguous secure and non-secure communications simultaneously.

DNP Implementations using TLS shall use the TCP port number 19999 by default to initiate secure connections.

DNP implementations using the application layer Secure Authentication mechanism but not TLS shall use port 20000 by default.

DNP implementations that do not use any security measures shall continue to use port 20000 by default as specified in Clause 13.

Implementations that use other than the default TCP port numbers for DNP shall be configurable to use the defaults.

7.7 Compliance with IEC/TS 62351-5

IEC/TS 62351-5 states that protocols claiming compliance to it shall include certain items in their specification. This subclause describes where in this specification those items are located.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.7.1 Selected options

Application layer authentication security in DNP3 is provided through an implementation of the IEC/TS 62351-5 standard. IEC/TS 62351-5 states:

— *The protocol specification shall identify which of the options identified in clause 8.3 are mandatory for the protocol (if any).*

— *The protocol specification shall identify any additional security algorithms, fixed parameters, configurable parameters or features supported by the protocol beyond the mandatory set specified in clause 8.2.*

The Device Profile Document for each DNP3 device supporting IEC/TS 62351-5 authentication shall identify this capability. Authentication is considered a subset of either master or outstation functionality to which a device may claim compliance, separate from any other subset.

All DNP3 devices shall support IEC/TS 62351-5 Aggressive Mode authentication in order to claim compliance to IEC/TS 62351-5 and DNP3. As noted in the specification, devices shall also permit aggressive mode to be disabled via configuration.

All DNP3 devices shall permit the Error Count to be configurable.

The Device Profile Document for DNP3 devices claiming compliance with IEC/TS 62351-5 shall also include the following information:

— A list of any and all hashing algorithms supported by the device in addition to the mandatory algorithms identified in IEC/TS 62351-5.

— A list of any and all encryption algorithms supported by the device addition to the mandatory algorithms identified in IEC/TS 62351-5.

7.7.2 Operations considered critical

IEC/TS 62351-5 states:

— *The protocol specification shall identify which protocol operations (e.g., function codes, ASDU types, control commands, setting changes) shall be considered Critical, requiring authentication through this mechanism.*

— *The protocol specification shall specify that certain operations described in 7.3.2.1.2 of this specification are always Critical.*

The mandatory and optional critical operations for DNP3 are specified in 7.3.2.1.2.

7.7.3 Addressing information

IEC/TS 62351-5 states:

— *The protocol specification shall identify which addressing information from the lower layers of the protocol shall be included in the HMAC calculation, as described in clauses 7.2.3.5 and 7.2.4.5, and the order of their octets in the calculation.*

DNP3 does not include any addressing information in the HMAC calculation, as described in the Data Object Library insert sheets for Group 120.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7.7.4 Message format mapping

IEC/TS 62351-5 states:

— *The protocol specification shall describe how each of the messages described in 7.7.2 shall be implemented using the protocol.*

— *The message formats described in the protocol specification message formats shall include all information found in the messages described in this specification.*

— *In general, the protocol specification shall use the sequence, layout, and naming of information described in this specification. The only exception to this requirement occurs if an equivalent piece of information already exists elsewhere in a protocol ASDU (such as a length parameter). Under such conditions the format and layout described in this specification may be altered. Such a parameter shall not be removed from the protocol entirely.*

— *The time stamp included in the Error message shall be in a format defined by the protocol. This format shall represent an unambiguous absolute time, not a relative time, e.g., "milliseconds since midnight on the following date…" is acceptable, but not "milliseconds since the previous midnight."*

— *The protocol specification shall not reduce the size or range of any information described in this specification.*

The mapping of message formats is described in 7.3.1 and the Data Object Library for Group 120. Notes there describe how the length of challenge data and some other fields is implemented using DNP3 qualifier codes. The time stamp used is six-octet DNP3 absolute time.

7.7.5 Reference to procedures

IEC/TS 62351-5 states:

— *The protocol specification shall specify how the procedures described in 7.7.3 shall be implemented using the protocol.*

— *If there is a disagreement between the procedures described in the protocol specification and the procedures described in this specification, this specification shall be deemed to be the correct description.*

The DNP3 procedures are described in 7.3.2 and are intended to be identical to those described in IEC/TS 62351-5.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

8. Transport function

8.1 Overview

8.1.1 Layering

The Transport Function is actually a sublayer of the Application Layer that fits at the bottom of the Application Layer just above the Data Link Layer (Figure 51). All messages to and from the Application Layer pass through the Transport Function on their way from and to the other station.

Figure 51—Transport Function location is between Application Layer and Data Link Layer

8.1.2 Purpose

The size of a DNP3 Application Layer message fragment may be larger than the number of octets permitted in a single Data Link Layer frame. The transport function disassembles Application Layer fragments into Data Link Layer-sized data units (called transport segments; see Figure 52) for transmission and reassembles these transport segments into the original application fragment on reception.

At the transmission site, an Application Layer fragment is broken into smaller portions, and a header is added containing sequencing information for each portion. The header and application data form a transport segment that is passed to the Data Link Layer. Transport segments are always passed one-at-a-time, in sequence, from the first to the last.

Header	Application Layer Data
← 1 octet →	← from 1 to 249 octets →

Figure 52—Transport segment

At the receiving site, transport segments passed from the Data Link Layer are checked for sequencing by examining the header. Any transport segments that are received out of order are discarded. Only when a complete application fragment is assembled does the transport function notify the Application Layer that it is available.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

8.2 Transport function description

8.2.1 Transport header

The transport header consists of a single octet. The header is the first octet in a transport segment. A header is prepended to each portion of Application Layer data before submission to the Data Link Layer for transmission. Upon receipt of a transport segment from the Data Link Layer, the header is stripped away before assembly of an application fragment.

The transport header octet is composed of three fields, as shown in Figure 53.

Bit #	7	6	5	4	3	2	1	0
Fields	FIN	FIR	SEQUENCE					

Figure 53—Header fields

8.2.1.1 FIN field

The FIN field is a single bit, which when set, indicates that this is the final or last transport segment in the fragment.

FIN = 0 indicates more transport segments follow.

FIN = 1 indicates the final transport segment in a series of transport segments.

8.2.1.2 FIR field

The FIR field is a single bit, which when set, indicates that this is the first transport segment in the fragment.

FIR = 0 indicates this is not the first transport segment in a series of transport segments.

FIR = 1 indicates this is the first transport segment in a series of transport segments.

8.2.1.3 SEQUENCE number field

The SEQUENCE number is a 6-bit field. It is used to verify that transport segments are received in the correct order and guards against duplicated or missing transport segments. It has a range of 0 to 63. Sequence numbers increment by one, modulo 64, for each transport segment in a series of transport segments that together hold a single Application Layer fragment. After sequence number 63, the next sequence number value is 0.

Each Application Layer fragment is transmitted as a series of consecutive transport segments.

8.2.1.4 Rules

— **Rule 1:** A transport segment-series may only begin with a transport segment having the FIR bit set.

— **Rule 2:** A transport segment-series ends with a transport segment having the FIN bit set.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— **Rule 3:** When a transport segment is received with the FIR bit set, any previously received, unterminated transport segment-series shall be discarded and a new transport segment-series shall be started with the newly received transport segment as its first member.

— **Rule 4:** When no transport segment-series is in progress, any transport segment received without the FIR bit set shall be discarded.

— **Rule 5:** A transport segment with the FIR bit set may have any sequence number from 0 to 63 without regard to prior history. Nevertheless, it is recommended that a transmitter increment the sequence number of all transport segments, including those with the FIR bit set, by one count (modulo 64) beyond the sequence number it placed in the previously transmitted transport segment.

— **Rule 6:** After a transport segment-series has been started:

1) Each subsequent received transport segment shall have a sequence number that is incremented by one (modulo 64) from the preceding transport segment.

2) A received transport segment that meets this requirement becomes the next member of the transport segment-series.

3) A received transport segment having the FIN bit cleared and a sequence number identical to the preceeding transport segment shall be discarded.

4) A received transport segment having the FIN bit set and a sequence number identical to the preceeding transport segment, or a received transport segment containing any sequence number other than the expected incremental number, shall be discarded and shall also cause the entire, in-progress transport segment-series to be terminated and discarded.

— **Rule 7:** A transport segment-series may consist of a single transport segment having both FIR and FIN bits set.

— **Rule 8:** When a complete transport segment-series is assembled, only then may its application data be passed to the Application Layer.

8.2.2 Application Layer Data

A transport segment consists of a transport header followed by 1 to 249 Application Layer data octets.

NOTE | The recommended practice for maximum efficiency is to use a transport segment size as large as possible for the communication environment.

8.2.2.1 Rules

— **Rule 1:** Each transport segment may contain 1 to 249 Application Layer data octets. Transport segments may contain fewer than 249 Application Layer data octets. It is not required that all segments, except the last, have the same size; it is permissible to vary the segment sizes within a segment-series. The receiver shall accept transport segments of varying sizes.

— **Rule 2:** The transport function preserves the octet order of the Application Layer fragment, as illustrated by the example in 8.2.3. At the receiving end, Application Layer fragments are reassembled in the same order.

1) Application data is divided into suitably sized portions or sets of octets.[28]

[28] The word *portion* is purposely used to avoid the term packet. The concepts are essentially the same; however, packet often infers the octets transmitted at the Physical Layer.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

2) The first application data portion, with octet number 0, is transported by the first transport segment; that segment has the FIR bit set in the header and a sequence number N.

3) The next application data portion, if there is one, is then placed into a transport segment having sequence number (N + 1) modulo 64.

4) This pattern, where application data portions with lower octet numbers are placed into transport segments before portions with higher octet numbers, continues until the last portion is sent.

5) The FIN bit set in the header of the last segment.

8.2.3 Segmenting Example

The following example illustrates the transport function.

EX 48	This example illustrates a 600-octet Application Layer fragment being divided into three portions and the organization of that data within transport segments. Octet numbers for the application data are shown inside the elements. The letters "TH" stand for Transport Header. Octets are transmitted in left-to-right order.

The Application Layer fragment is divided into three portions.

0	1	…	247	248	249	250	…	496	497	498	499	…	598	599
← 1st portion →					← 2nd portion →					← 3rd portion →				

First transport segment:

	0	1	…	247	248
TH	← 1st application data portion →				

Transport header: FIR = 1, FIN = 0, SEQUENCE = *n* (any number 0–63, e.g.: 25).

Second transport segment:

	249	250	…	496	497
TH	← 2nd application data portion →				

Transport header: FIR = 0, FIN = 0, SEQUENCE = (*n*+1) modulo 64 (e.g.: 26).

Third transport segment:

	498	499	…	598	599
TH	← 3rd application data portion →				

Transport header: FIR = 0, FIN = 1, SEQUENCE = (*n*+2) modulo 64 (e.g.: 27).

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

8.2.4 Reception state table

Assume the transport function software has a fragment buffer where application data from the received transport segments are temporarily stored before presenting the fragment to the Application Layer.

The software requires two states for proper operation.

a) Idle state: The software is idle waiting for a transport segment to arrive with the FIR bit set.

b) Assembly state: The fragment buffer holds application data from at least one transport segment. While in this state, the software is awaiting additional transport segments to complete the fragment.

Keys for understanding the table:

— The events referred to in italics in the table heading are triggers that initiate an action and possible transition to a different state. They do not refer to the Application Layer buffered events or event objects transported in messages.

— X means don't care.

— SAME means the sequence number is identical to the sequence number in the transport segment immediately preceeding this transport segment.

— +1 means the sequence number is incremented by one count, modulo 64, from the sequence number in the transport segment immediately preceeding this transport segment.

— +M, $1 < M < 64$ means the sequence number is incremented by more than one count and less than 64 counts from the sequence number in the transport segment immediately preceeding this transport segment.

— Read Table 46 as follows:

If the software is currently in the state shown in column A,

and a transport segment is received with the fields in its Transport Header as shown in column C,

then perform the action stated in column D,

and go into the software state specified in column E.

Column B contains comments regarding the received transport segment.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 46—Transport function reception table

Current state	Event that triggers an action and possible transition				Action	Transition to state	
A	B	C			D	E	
If the software state is	and a transport segment is received (comments appear below)	with these fields in its transport header			then perform this action	and go to this state	
		FIR	FIN	SEQ			
Idle	Not a first segment.	0	X	X	Discard transport segment.	Idle	1
	Entire application data fits within the segment.	1	1	X	Clear the fragment buffer, place transport segment's data into the fragment buffer and pass fragment buffer to Application Layer.	Idle	2
	First of multiple segments.	1	0	X	Clear the fragment buffer and place transport segment's data into the fragment buffer.	Assembly	3
Assembly	Same SEQ as previous, more segments are expected.	0	0	SAME	Discard transport segment.	Assembly	4
	Same SEQ as previous, final segment received.	0	1	SAME	Discard transport segment.	Idle	5
	Expected segment received, more segments are expected.	0	0	+1	Append transport segment's data to contents of fragment buffer.	Assembly	6
	Expected segment received, final segment.	0	1	+1	Append transport segment's data to contents of fragment buffer and pass fragment buffer to Application Layer.	Idle	7
	SEQ is out of order.	0	X	+M 1 < M < 64	Discard transport segment.	Idle	8
	First of multiple segments.	1	0	X	Clear contents of fragment buffer and place transport segment's data into the fragment buffer.	Assembly	9
	Entire application data fits within the segment.	1	1	X	Clear contents of fragment buffer, place transport segment's data into the fragment buffer and pass fragment buffer to Application Layer.	Idle	10

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

8.2.5 Reception state diagram

The text at the beginning of 8.2.4 regarding fragment buffers and the two states applies to the diagram in Figure 54. The term "segment" in the diagram refers to a transport segment.

Figure 54—Reception state diagram

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

9. Data Link Layer

9.1 Overview

9.1.1 Layering

The Data Link Layer provides an interface between the Transport Function sub-layer of the Application Layer and the physical media or network Transport Layer. The main contributions of the Data Link Layer are station addressing and error detection. This layer adds the last DNP3-specific overhead octets when transmitting over the communication channel.

The protocol has been designed to operate in a byte-oriented mode over a communications channel, as well as over packet-oriented networks using TCP/IP and UDP/IP.

The DNP3 Data Link Layer is suitable for both connection-less and connection-oriented systems. Connection-oriented infers physical networks that require dialing, logging in, or otherwise establishing a communication channel before data transfer to the destination device can transpire. This document does not include connection service requirements. These requirements are system dependent and beyond the scope of the DNP3 protocol (Figure 55).

Figure 55—DNP3 protocol stack

9.2 DNP3 Data Link Layer description

9.2.1 Introduction

In DNP3, the Data Link Layer has two main purposes. First, it bi-directionally transports Application Layer data across the communication channel to the destination device. The Data Link Layer software encodes the transport function segments passed down from above within a data link frame and sends the frame to the communication channel for transmission. Conversely, transport segments contained within a received data link frame are extracted and passed up to the higher layer.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Second, the Data Link Layer manages the variables associated with data link frame synchronization and error control, and it provides indications of link status.

9.2.2 Services

The Data Link Layer provides the following services to the overall protocol:

— Encapsulation of transport segments into the data link frame for transmission over the communication channel.

— Decoding of data link frames received from the communication channel into transport segments.

— Error detection. Each transmitted frame contains a CRC word for every 16 data octets and one for the data link frame header. These CRC words are checked in each received frame so that only valid data are passed to the Transport Function.

— Source and destination addressing. DNP3 requires both addresses to enable multiple masters and multiple outstations to share the communication channel.

— Optional confirmation of each data link frame.

— Optional detection of lost or repeated data link frame requests.

9.2.3 Transaction model

The Data Link Layer uses transactions for conveying data to the Data Link Layer in another DNP3 device and to perform link management functions (Figure 56). A transaction consists of one or two messages.

a) A request message from the device initiating the transaction, called the primary station, to another device, called the secondary station.

b) Optionally, a response message from the secondary station back to the primary station.

NOTE—Data Link Layer requests and responses are distinct from the requests and responses in the Application Layer.

Request messages, from the primary to secondary, always contain a command for the secondary. A function code is included in the message to specify the command. Some of the function codes are intended for managing the link. Here the message contents are private between both Data Link Layers and are not conveyed to the higher layers. Other commands are used for conveying data from the layer above the Data Link Layer in the primary station to the layer above the Data Link Layer in the secondary station.

Some, but not all, of the Data Link Layer function codes in the primary-to-secondary messages require a response. The response may simply acknowledge the receipt of the primary message, or it may contain information about the secondary's Data Link Layer, but it never contains data from the higher layers.

The main job of the Data Link Layer is to carry data from upper layers called "user data" (data from the user of the Data Link Layer). The Data Link Layer always transmits user data in a primary-to-secondary request message.

The Data Link Layer in each DNP3 device acts as both a primary and as a secondary station. The behavior as a secondary is independent of its behavior as a primary. For example, when a master station issues a poll, its Application Layer passes information to the Data Link Layer requesting it to initiate a transaction to send the information to the Application Layer in an outstation. The master is the primary, and the outstation is the secondary. When the Application Layer in the outstation has data to return to the master, it passes that information to its Data Link Layer requesting it to initiate a transaction to send the data to the

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Application Layer in the master. In this instance, the outstation is the primary and the master is the secondary.

Figure 56—Transaction diagram

9.2.4 Frame format

This subclause describes the DNP3 data link frame format (Figure 57). A data link frame has a fixed length header block, block 0, followed by optional data blocks. Each block ends with a 16-bit CRC.

Figure 57—DNP3 frame format

9.2.4.1 Data Link Layer header frame fields

This subclause describes Block 0, or header, of a data link frame. The header fields consist of 2 start octets, 1 length octet, 1 link control octet, a two-octet destination address, and a two-octet source address.

9.2.4.1.1 START field

The START field is 2 octets in length. The first octet is a 0x05, and the second octet is a 0x64.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

9.2.4.1.2 LENGTH field

The LENGTH field is 1 octet in length and specifies the count of non-CRC octets that follow in the header and data blocks. This count includes the CONTROL, DESTINATION, and SOURCE fields in the header and the USER DATA fields in the body. CRC fields are not included in the count. The minimum value for this field is 5, indicating only the header is present, and the maximum value is 255.

9.2.4.1.3 CONTROL field

The CONTROL field is 1 octet in length and contains information about the frame's direction, transaction initiator, error and flow control, and function.

Figure 58 shows the control octet fields.

<table>
<tr><td rowspan="2">DIR</td><td rowspan="2">1
PRM
0</td><td>FCB</td><td>FCV</td><td colspan="4" rowspan="2">FUNCTION CODE</td><td>← Primary to secondary</td></tr>
<tr><td>0</td><td>DFC</td><td>← Secondary to primary</td></tr>
<tr><td>Bit 7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td></td></tr>
</table>

Figure 58—Control octet bit definitions

9.2.4.1.3.1 DIR bit field

The Direction bit indicates the physical transmission origin of the data link frame.

DIR = 1 indicates a frame from a Master.

DIR = 0 indicates a frame from an Outstation.

NOTE—Transmitting devices shall set this bit correctly, but receiving devices are not obligated to check its state.

9.2.4.1.3.2 PRM bit field

The Primary Message Bit indicates the direction of the data link frame with respect to the initiating station.

PRM = 1 indicates a Data Link Layer transaction is being initiated by either a master or an outstation, and the function code is chosen from Table 47. The FCV and FCB fields apply.

PRM = 0 indicates a Data Link Layer transaction is being completed by either a master or an outstation, and the function code is chosen from Table 48. The DFC field applies.

9.2.4.1.3.3 FCB bit field

The Frame Count bit (FCB) is used to detect losses and duplication in primary-to-secondary frames. The Data Link Layer in a secondary station can detect that a request is new when it receives a frame having the FCV bit set and an FCB that matches the state expected by the secondary. When the secondary detects this condition, it toggles the state that it expects for the FCB in the next message with the FCV bit set. If the received FCB bit in a message does not match the expected FCB state when the FCV is set, the secondary recognizes the message as being a repeat of a previous message and acts accordingly. The primary station toggles its FCB upon receiving a frame with the ACK function code back from the secondary.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

The FCB is only valid in Data Link Layer request messages sent from primary to secondary with the FCV bit set. Table 47 specifies which particular messages may set the FCV bit.

9.2.4.1.3.4 FCV bit field

The Frame Count Valid (FCV) bit appears in every primary-to-secondary frame and specifies whether the secondary is to examine the FCB bit.

> FCV = 1 indicates the state of the FCB bit is valid and that the state of the FCB bit in the received message shall be checked against its expected state.
>
> FCV = 0 indicates the state of the FCB bit is ignored.

Before a station acting as a primary can transmit primary-to-secondary frames with the FCV bit set, it shall first verify that the secondary station's expected FCB (EFCB) is properly synchronized. The primary does this by sending a request to reset the secondary's link states. It should send this message upon detection of communication loss or whenever either station restarts.

Each secondary station, after Data Link Layer startup or transaction failure, shall **not** accept any primary station request messages having the FCV bit set until its FCB state has been reset by receiving a reset link states request from the primary.

9.2.4.1.3.5 DFC bit field

The Data Flow Control (DFC) bit appears in every response from a secondary station regardless of the function code in the control octet. It is used to report an insufficient number of Data Link Layer buffers to hold a receive frame. It is also used to indicate that the secondary's Data Link Layer is busy.

> DFC = 1 indicates receive buffers were not available or that secondary's Data Link Layer was busy.
>
> DFC = 0 indicates receive buffers were available and the secondary's Data Link Layer was ready.

9.2.4.1.3.6 FUNCTION CODE field

The Function Code identifies the function or service associated with the data link frame. The definition of the values placed in this field depends on whether Data Link Layer messages are sent from the primary station to the secondary station or from the secondary station to the primary station. Table 47 and Table 48 specify the codes and associated FCV states. A detailed description of each function is provided in 9.2.6 and 9.2.7.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 47—Primary-to-secondary (PRM = 1) function codes

Primary function code	Function code name	Service function	FCV bit	Response function codes permitted from secondary
0	RESET_LINK_STATES	Reset of remote link	0	0 or 1
1	—	Obsolete	—	15 or no response
2	TEST_LINK_STATES	Test function for link	1	0 or 1 (no response is acceptable if the link states are UnReset)
3	CONFIRMED_USER_DATA	Deliver application data, confirmation requested	1	0 or 1
4	UNCONFIRMED_USER_DATA	Deliver application data, confirmation not requested	0	No response
5	—	Reserved	—	15 or no response
6	—	Reserved	—	15 or no response
7	—	Reserved	—	15 or no response
8	—	Reserved	—	15 or no response
9	REQUEST_LINK_STATUS	Request status of link	0	11
10	—	Reserved	—	15 or no response
11	—	Reserved	—	15 or no response
12	—	Reserved	—	15 or no response
13	—	Reserved	—	15 or no response
14	—	Reserved	—	15 or no response
15	—	Reserved	—	15 or no response

Table 48—Secondary-to-primary (PRM = 0) function codes

Secondary function code	Function code name	Service function
0	ACK	Positive acknowledgment
1	NACK	Negative acknowledgment
2	—	Reserved
3	—	Reserved
4	—	Reserved
5	—	Reserved
6	—	Reserved
7	—	Reserved
8	—	Reserved
9	—	Reserved
10	—	Reserved
11	LINK_STATUS	Status of the link
12	—	Reserved
13	—	Reserved
14	—	Obsolete
15	NOT_SUPPORTED	Link service not supported

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

All devices shall accept and process messages received with primary-to-secondary function codes 0, 2, 3, 4, and 9. It is not permissible to reply to these messages with secondary-to-primary function code 15.

A secondary station shall not reply to any request that contains a destination broadcast address regardless of which primary-to-secondary function code appears in the request. A primary station, when transmitting broadcast messages should only use primary-to-secondary function code UNCONFIRMED_USER_DATA.

9.2.4.1.4 DESTINATION field

The Destination address field is two octets in size and specifies the address of the station to which the frame is directed. The first octet of the address is the low order octet, and the second octet is the high order (Figure 59).

Low Order Octet (LSB)	High Order Octet (MSB)

Figure 59—Destination address format

Refer to 9.2.5 for more information regarding addresses.

9.2.4.1.5 SOURCE field

The Source Address field is two octets in size and specifies the address of the station from where the frame originates. The first octet of the address is the low order octet, and the second octet is the high order (Figure 60).

Low Order Octet (LSB)	High Order Octet (MSB)

Figure 60—Source address format

Refer to 9.2.5 for more information regarding addresses.

9.2.4.2 User data

User data refers to the link user's data; in other words, the data belonging to the higher layers that use the Data Link Layer. One philosophy behind the link layer design is that of layer independence. The link layer provides a service to the higher layers but does not need to know what are the contents of the data that it transports.

Referring to Figure 57 the blocks following the header block hold the link user's data (also known as the payload). The maximum number of user data octets that a single frame can hold is 250. (5 octets in the link header are included in the octet count, thus leaving a maximum 250 for user data.) Each block, except the last user data block, shall contain exactly 16 octets of link user's data. The last block holds the remaining octets if the total number of user data octets is not evenly divisible by 16. Each user data block has a 2-octet CRC appended to it.

Not all frames convey user data. Some are used for managing the link.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

9.2.4.3 CRC fields

A 2-octet cyclic redundancy check is appended to each block in a frame. The START, LENGTH, CONTROL, DESTINATION, and SOURCE fields are all included when calculating the CRC for the header.

The 2-octet CRC check is generated from the following polynomial and then inverted before being placed in the block for transmission:

$$X^{16} + X^{13} + X^{12} + X^{11} + X^{10} + X^{8} + X^{6} + X^{5} + X^{2} + 1$$

Subclause 9.4 provides methods for computing this CRC.

Figure 61 shows the ordering of the 2-octet CRC check word within each header or user data block.

first octet … last octet	CRC	
Header or user data octets	LSB	MSB

Figure 61—CRC ordering

EX 49	This example shows the CRC for a Test Link States frame from a primary station with address 0 to secondary station with address 1.

►►► Request Message

05	64	05	F2	01	00	00	00	52	0C
START		LEN	CTRL	DESTINATION		SOURCE		CRC	

9.2.4.4 Inter-octet and inter-frame gaps

Each Data Link Layer frame should be transmitted without inter-octet gaps.

NOTE Receivers may optionally choose to discard an entire frame and begin waiting for start octets if they detect a gap in the frame. However, this is a processing efficiency concern because it is possible for receivers to recover from errors without such a feature. DNP3 does not specify the timing for this option.

In DNP3, a device is permitted to transmit two messages consecutively without an interval between them. Receiving devices shall not require an inter-frame gap to identify new frames. Frames shall pass the tests enumerated in 9.2.9.

Clause 10 of this specification contains more information on this subject.

9.2.5 Addressing notes

9.2.5.1 Choosing addresses

Each master and outstation device on a single link shall utilize a DNP3 address in the range 0x0000 through 0xFFEF that is unique from all other devices on the link.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

9.2.5.2 Reserved and special use addresses

Addresses in the range 0xFFF0 through 0xFFFF are reserved by DNP3 for special use, and devices shall not use these for their individual address assignment (Table 49).

Table 49—Special use addresses

Address	Special use
0xFFFF	All-call, Application Layer Confirmation to clear IIN1.0 is optional
0xFFFE	All-call, Application Layer Confirmation to clear IIN1.0 is mandatory
0xFFFD	All-call, Application Layer Confirmation shall not be required to clear IIN1.0
0xFFFC	Self-address
0xFFF0 to 0xFFFB	Reserved

9.2.5.2.1 All Stations Addresses

Addresses 0xFFFD to 0xFFFF are defined as broadcast or all-call addresses, and all stations shall accept frames with the destination address set to these values. See Clause 4 through Clause 6 for more information regarding behavior specific to these three addresses. Messages transmitted to these addresses shall never use the confirmed delivery function code CONFIRMED_USER_DATA.

NOTE The Data Link Layer should be able to pass to higher layers information regarding which address is received when an all-stations broadcast message is received. This is because the Application Layer is obligated to behave in specific ways depending on which destination address was received.

9.2.5.2.2 Self-address

Address 0xFFFC is called the "Self-address," and it shall only appear in the destination address field. Support for it is optional. Devices that support this address, and have the self-address feature enabled, shall process frames with destination address 0xFFFC as if the message had used the device's unique individual address. Responses always include the device's individual address in the source address field. This feature can simplify the commissioning, troubleshooting and maintenance of devices because it is not necessary to know the receiving device's address ahead of time. Once the destination device's true address is discovered, testing can resume with the proper individual address.

Devices that support the self-address shall have a means to disable it.

NOTE Only enable a single device at a time for processing messages with the self-address destination so that multiple devices do not respond.

9.2.5.2.3 DNP3 reserved addresses

Addresses 0xFFF0 to 0xFFFB are reserved for future use by DNP3, and users shall not assign these for any reason. Messages addressed to these addresses do not need to be passed to the higher layers.

9.2.6 Primary-to-secondary function codes

These function codes apply when the PRM field is a 1 in the header block control octet.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

9.2.6.1 Function code 0 RESET_LINK_STATES

This function code is utilized for synchronizing a secondary station's states so that it properly processes primary-to-secondary frames with the FCV bit set. This function code is only required if data is transmitted using function code CONFIRMED_USER_DATA.

A DNP3 device shall treat every other device that it communicates with independently. The following applies to each primary–secondary station pair. The states and messages between a pair of stations are unrelated to the states and messages between any other pair.

A secondary station may not pass confirmed user data (received with primary-to-secondary function code CONFIRMED_USER_DATA) to the higher layers until it has been synchronized from the primary station.

The primary station shall send this function code to the secondary prior to communicating using confirmed user data requests. The primary should perform the synchronization after it starts up (reboots) or after recovery from a communications failure with the secondary.

When a secondary station re-starts or if it loses communication with the primary, the secondary's link states are unsynchronized. The secondary station is then considered to be in an UnReset state and may not pass confirmed user data to the higher layers.

This function code synchronizes the FCB bit between primary and secondary stations. After completing the reset link states transaction, a secondary shall expect the FCB bit to be 1 in the next message with the FCV bit set. Similarly, after completing the reset link states transaction, a primary station shall set the FCB bit to 1 in the next message it sends with the FCV bit set.

When a device X transmits a reset link states message to device Y, X is the primary and Y is the secondary. Confirmed communications are only enabled in one direction, from X to Y. Before Y can send confirmed communications to X, Y shall transmit a reset link states request to X. Both X and Y behave as primary and as secondary depending on what services are required by the higher layers.

9.2.6.2 Function code 2 TEST_LINK_STATES

The test link states request is utilized for testing the states of the secondary's Data Link Layer. Upon reception of this function code by a secondary station, it checks the value of the FCB bit in the primary message and compares it against the FCB state it expects from that primary station.

— If the FCBs match, the secondary should send an ACK confirm response frame and toggle the expected FCB state from that primary. The secondary station sets the DFC bit accordingly in the response.

— If the FCBs do **not** match, then the secondary station shall send a repeat of the most recent ACK or NACK response frame that it transmitted.

— If the secondary receives a test link states request without there having been a previous link reset states request, it shall either not respond or respond with a NACK with DFC = 0.

9.2.6.3 Function code 3 CONFIRMED_USER_DATA

This function code is utilized for sending user data to a secondary station with the requirement that the secondary return confirmation of the data's arrival. The frame contains the data provided from the primary's upper layers (Data Link Layer user). At the secondary, the data from the received frame are passed to the secondary's user. Before this service can succeed, the secondary station's link states shall be reset as described in 9.2.6.1.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Upon reception of this function code by a secondary station, it checks the value of the FCB bit in the primary message and compares it against the FCB state it expects from that primary station.

— If the FCBs match, the secondary should send an ACK confirm response frame, pass the user data to the higher layers, and toggle the expected FCB state from that primary. The secondary station sets the DFC bit accordingly in the response.

— If the FCBs do **not** match, the secondary should send an ACK confirm response frame, but it does **not** pass the data to the higher layers, and it does **not** toggle the expected FCB state from that primary. The secondary station sets the DFC bit accordingly in the response.

NOTE Do not use CONFIRMED_USER_DATA when using TCP or UDP over an IP network,

9.2.6.4 Function code 4 UNCONFIRMED_USER_DATA

This function code is utilized for sending user data to the secondary station without requiring the secondary to return confirmation of the data's arrival. The frame contains the data provided from the primary's upper layers (Data Link Layer user). At the secondary, the data from the received frame are passed to the secondary's user.

This function reduces bandwidth requirements because fewer messages are transmitted. This is the preferred function code for sending broadcast application data, or sending application data when there is another means of confirming reception.

9.2.6.5 Function code 9 REQUEST_LINK_STATUS

The request link status function is utilized to obtain the status of the secondary Data Link Layer. The primary may send this function at any time. This function can be used for "Are you there?" and keep-alive messages when the environment calls for knowing that the other end is still connected or on-line.

NOTE A message with the REQUEST_LINK_STATUS function code is recommended for verifying that the other station is still connected, and on-line, when DNP3 is used over IP networks.

9.2.7 Secondary-To-primary function codes

These function codes apply when the PRM field is 0 in the header block control octet. The Data Link Layer responses that return these function codes never include user data blocks. Refer to Figure 57.

NOTE—The DFC bit is always set to an appropriate value (see 9.2.4.1.3.5) in every secondary response.

9.2.7.1 Function code 0 ACK

The secondary station returns an ACK response as a positive confirmation that the intended request was accepted from the primary. An ACK response is only applicable to request function codes RESET_LINK_STATES, TEST_LINK_STATES, and CONFIRMED_USER_DATA.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

9.2.7.2 Function code 1 NACK

The secondary station returns a NACK response as a negative confirmation that the request is not accepted from the primary for one of these reasons:

— The secondary's link state is not reset when either a primary-to-secondary function TEST_LINK_STATES or CONFIRMED_USER_DATA frame is received with the FCV bit set.

— The secondary's link is busy.

— The secondary's Data Link Layer receive buffers are full.

A NACK response is only applicable to request function codes RESET_LINK_STATES, TEST_LINK_STATES, and CONFIRMED_USER_DATA.

9.2.7.3 Function code 11 LINK_STATUS

The link status response shall be returned from the secondary when it receives a request link status message from the primary.

When the primary requests the link status as an "Are you there?" or keep-alive message, the response, which requires this function code, serves as an indication that the secondary did receive the request.

9.2.7.4 Function code 15 NOT_SUPPORTED

The function is utilized in responses returned by the secondary when the primary requests a function code that is not supported or has not been implemented.

9.2.8 Link control variables

Every device shall maintain a separate set of variables for every other device with which it communicates using confirmed link layer services. The minimum sets are shown in 9.2.8.1 and 9.2.8.2.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

9.2.8.1 Primary station variables

Variable	Description
SecondaryStationIsReset	This is a boolean variable that indicates that the primary station sent a RESET_LINK_STATES frame to the secondary and received an ACK confirmation back.
SoftwareOperatingState	This variable indicates the current state in which the software is operating. The states are shown in column A of Table 50.
NFCB (Next FCB)	The 1 or 0 state of the next FCB bit to be included when transmitting a new primary-to-secondary frame with the FCV bit set.

9.2.8.2 Secondary station variables

Variable	Description
LinkIsReset	This is a boolean variable that indicates whether the link was reset via receipt of a RESET_LINK_STATES frame from the primary station.
SoftwareOperatingState	This variable indicates the current state in which the software is operating. The states are shown in column A of Table 51.
EFCB (Expected FCB)	The state of the FCB bit expected in the next frame from the primary station with the FCV bit set.

9.2.9 Frame error detection

The receiving device's Data Link Layer shall inspect frames before acceptance. Software should examine the following, and if any of these conditions are false, discard the frame.

— The starting octets shall be 0x05 followed by 0x64; see 9.2.4.1.1.

— The destination address shall match one of the addresses that the Data Link Layer is programmed to accept; see 9.2.4.1.4. Addresses include the reserved and special addresses in 9.2.5.2.

— The source address shall match one of the addresses that the Data Link Layer is programmed to accept; see 9.2.4.1.5. A device may optionally choose to accept frames from any address.

— Correct CRC values shall appear at the designated locations; see 9.2.4 and 9.2.4.3.

— The frame length shall match the value in the length field; see 9.2.4.1.2.

— Function codes shall comply with those specified for the primary-to-secondary requests in Table 47 and the secondary-to-primary responses in Table 48.

— The FCV bit shall be set in primary-to-secondary frames, depending on the function code, according to Table 47.

9.2.10 Collision avoidance

Operation in some systems requires a device to detect whether its transmission would occur while another device is transmitting. If this were to happen, the transmissions "collide" and almost certainly cause communication errors at the listening device or devices. In order to prevent clashing, a device that detects

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

another's transmission is in progress shall delay its pending message communication until no such collision is likely to occur.

Devices detect collisions in one of two ways.

a) Detecting carrier existence in the physical layer.

b) Detecting reception of data octets.

NOTE

DNP3 does not specify the physical layer. Nevertheless, many communications are based on the EIA-232-F standard, and for those physical layers that do have carriers (modems and radios are an example), the DCD signal may provide an indication of carrier presence.

Other physical layers use base-band techniques (the EIA-485-A standard and fiber optics are examples) and do not have a carrier. A technique for detecting the absence of another device transmitting is to wait until no octets have been received for a configurable amount of time.

A device that is ready to transmit should wait for a hold-off delay time after detecting no other device is transmitting. The hold-off delay should consist of two parts added together—a configurable fixed time and a random time. These times are system dependent and not specified by DNP3. The hold-off timing begins when no other device transmissions are detected; this instant may begin prior to the device having a message fully ready for transmission.

NOTE

The fixed delay can be adjusted per device to provide priority of access. For instance, it can be zero for masters and non-zero for outstation devices. The fixed delay for the outstations shall be greater than the maximum random delay of the master. This provides a minimum window of access time for the master. To verify the master's priority, the master does not add in a random delay.

Many systems do not require collision avoidance, and its implementation is optional.

9.3 State tables and diagrams

9.3.1 Primary station state requirements

9.3.1.1 Explanatory statements

This subclause describes the state requirements for a primary station that transmits a request to another device, the secondary.

Keys for understanding Table 50 and Figure 62:

— The events referred to in italics in the table heading are triggers that initiate an action and possible transition to a different state. They do not refer to the Application Layer buffered events or event objects transported in messages.

— Columns in the table are identified by the letters A through E.

— Rows in the table are numbered from 1 to 35.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— The circles in the figure represent states, and the lines represent transitions to a new state. The numbers on the lines correspond to the respective row in the table.

— Read the table as follows:

— The software waits in one of the states shown in column A until an event initiates further action.

The cause for action (event) is described in column B.

The action performed is described in columns C and D. The action in column C, if any, is performed before sending a message with the function code specified in column D.

After performing the actions in columns C and D, the software transitions to the state specified in column E. Column E specifies one of the states listed under column A.

— NFCB means **N**ext **F**rame **C**ount **B**it. It is the 1 or 0 state of the next FCB bit to be included when transmitting a new primary-to-secondary frame with the FCV bit set.

— The ▲ symbol indicates that variable SecondaryStationIsReset is set to true.

— The ▼ symbol indicates that variable SecondaryStationIsReset is set to false.

— Except for rows 27 and 28, the table does not show the behavior when the secondary response sets the DFC bit (see 9.2.4.1.3.5). If this happens, the primary station has the option of

1) Waiting a reasonable time before transmitting user data.
2) Sending a REQUEST_LINK_STATES message and then proceeding to the
   i) UR-LinkStatusWait state if variable SecondaryStationIsReset is set to false.
   ii) R-LinkStatusWait state if state variable SecondaryStationIsReset is set to true.
3) Setting variable SecondaryStationIsReset to false and proceeding to the SecUnResetIdle state.

— Where “Implementation dependent” or “Implementation dep.” appears in the table, it means that the cause for action is a private reason or that the action to be performed is particular to the software design.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

9.3.1.2 Primary station state table

Table 50—Primary action state table

Current state	Event that triggers an action and possible transition	Action			Transition to state	
A	B	C	D		E	
If the software state is	Cause for action	Perform this action	and transmit this function code (with user data if appropriate)		and then go to this state	
SecUnResetIdle	Implementation dependent	Set retry count = 0	0	RESET_LINK_STATES	ResetLinkWait-1	1
	User request to send confirmed data	Set retry count = 0	0	RESET_LINK_STATES	ResetLinkWait-2	2
	User request to send unconfirmed data	—	4	UNCONFIRMED_USER_DATA	SecUnResetIdle	3
	Keep-alive or implementation dep.	Set retry count = 0	9	REQUEST_LINK_STATUS	UR–LinkStatusWait	4
SecResetIdle	Implementation dependent	Set retry count = 0	0	RESET_LINK_STATES	ResetLinkWait-1	5
	User request to test link	Set retry count = 0	2	TEST_LINK_STATES	TestWait	6
	User request to send confirmed data	Set retry count = 0	3	CONFIRMED_USER_DATA	CfmdDataWait	7
	User request to send unconfirmed data	—	4	UNCONFIRMED_USER_DATA	SecResetIdle	8
	Keep-alive or implementation dep.	Set retry count = 0	9	REQUEST_LINK_STATUS	R–LinkStatusWait	9
ResetLinkWait-1	Function code 0 (ACK) received	▲ Set NFCB = 1.	—		SecResetIdle	10
	Non-0 function code received	▼	—		SecUnResetIdle	11
	Timeout, no response, no retry	▼ Inform user of failure	—		SecUnResetIdle	12
	Timeout, no response, retry	Increment retry count	0	RESET_LINK_STATES	ResetLinkWait-1	13
ResetLinkWait-2	Function code 0 (ACK) received	▲ Set NFCB = 1.	3	CONFIRMED_USER_DATA	CfmdDataWait	14
	Non-0 function code received	▼	—		SecUnResetIdle	15
	Timeout, no response, no retry	▼ Inform user of failure	—		SecUnResetIdle	16
	Timeout, no response, retry	Increment retry count	0	RESET_LINK_STATES	ResetLinkWait-2	17
UR–LinkStatusWait	Function code 11 (LINK_STATUS) received	Implementation dependent	—		SecUnResetIdle	18

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Current state	Event that triggers an action and possible transition	Action		Transition to state	
A	B	C	D	E	
If the software state is	Cause for action	Perform this action	and transmit this function code (with user data if appropriate)	and then go to this state	
	Non-11 function code received	Note erroneous response	—	SecUnResetIdle	19
	Timeout, no response, no retry	Inform user of failure	—	SecUnResetIdle	20
	Timeout, no response, retry	Increment retry count	9 REQUEST_LINK_STATUS	UR–LinkStatusWait	21
TestWait	Function code 0 (ACK) received	Toggle NFCB.	—	SecResetIdle	22
	Non-0 function code received	▼	—	SecUnResetIdle	23
	Timeout, no response, no retry	▼ Inform user of failure	—	SecUnResetIdle	24
	Timeout, no response, retry	Increment retry count	2 TEST_LINK_STATES	TestWait	25
CfmdDataWait	Function code 0 (ACK) received	Toggle NFCB.	—	SecResetIdle	26
	Function code 1 (NACK) received, DFC = 0	▼ Set retry count = 0.	0 RESET_LINK_STATES	ResetLinkWait-2	27
	Function code 1 (NACK) received, DFC = 1	▼	—	SecUnResetIdle	28
	Non-0, non-1 function code received	▼	—	SecUnResetIdle	29
	Timeout, no response, no retry	▼ Inform user of failure	—	SecUnResetIdle	30
	Timeout, no response, retry	Increment retry count	3 CONFIRMED_USER_DATA	CfmdDataWait	31
R–LinkStatusWait	Function code 11 (LINK_STATUS) received	Implementation dependent	—	SecResetIdle	32
	Non-11 function code received	Note erroneous response	—	SecResetIdle	33
	Timeout, no response, no retry	▼ Inform user of failure	—	SecUnResetIdle	34
	Timeout, no response, retry	Increment retry count	9 REQUEST_LINK_STATUS	R–LinkStatusWait	35

NOTE Table 50 does not show potential error conditions such as misbehaving secondary stations. Implementers are encouraged to incorporate defensive programming techniques to prevent endless looping and other problems.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

9.3.1.3 Primary station state diagram

Figure 62—Primary station state diagram

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

9.3.2 Secondary station state requirements

9.3.2.1 Explanatory statements

This subclause describes the state requirements when a device is a secondary and receives a message from another device with the PRI bit set to 1.

The software requires two states for proper operation.

a) UnReset state: The secondary was re-started, power was restored, or a communication loss with the primary was detected.

b) Idle state: The software is idle waiting for a message to arrive from the primary.

The correct DFC state is included in the link control octet of every response returned to the primary. It is not shown in the table.

If the PRI bit is 0 in the received request, no action is taken and no response is transmitted. The PRI bit is always set to 0 in responses sent to the primary. These bits are not shown in the table.

Keys for understanding Table 51 and Figure 63:

— The events referred to in italics in the table heading are triggers that initiate an action and possible transition to a different state. They do not refer to the Application Layer buffered events or event objects transported in messages.

— Columns in the table are identified by the letters A through E.

— Rows in the table are numbered from 1 to 24.

— The circles in the figure represent states, and the lines represent transitions to a new state. The numbers on the lines correspond to the respective row in the table.

— The values in the “Func” columns refer to the function code numbers in the respective received or transmitted link control octet.

— X means don’t care.

— .. (double dot) means consecutively through, or the entire range beginning with the number on the left through the number on the right.

— EFCB means **E**xpected **F**rame **C**ount **B**it. It is the state of the FCB bit expected in the next frame from the primary station with the FCV bit set.

— == means is the same as.

— != means is not the same as.

— The ▲ symbol indicates that variable LinkIsReset is set to true.

— Read the table as follows:

If the software is currently waiting in the state shown in column A,

and a frame is received from the primary with the control octet fields as shown in column B,

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

then perform the action stated in column C,

then send a reply frame back to the primary with a function code in the control octet as shown in column D,

and go into the reception state specified in column E. Column E specifies one of the states listed under column A.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

9.3.2.2 Secondary station state table

Table 51 —Secondary station state table

Current state	Event that triggers an action and possible transition			Action			Transition to state	
A	B			C	D		E	
If the software state is	and a frame is received with link control octet fields			then perform this action	and send this reply		and go to this state	
	FCB	FCV	Func		Func	Comments		
UnReset	X	0	0	▲ Set EFCB = 1	0	Ack.	Idle	1
	X	1		Do nothing.	15	May optionally not transmit anything.	UnReset	2
	X	X	1,5..8, 10..15	Do nothing.	15	May optionally not transmit anything.	UnReset	3
	X	0	2	Do nothing.	15	May optionally not transmit anything.	UnReset	4
	X	1		Do nothing.	1	May optionally not transmit anything.	UnReset	5
	X	0	3	Do nothing.	15	May optionally not transmit anything.	UnReset	6
	X	1		Do nothing.	1	May optionally not transmit anything.	UnReset	7
	X	0	4	Send request to application parsing routine.	Do not transmit anything		UnReset	8
	X	1		Do nothing.			UnReset	9
	X	0	9	Do nothing.	11	Status of link.	UnReset	10
	X	1		Do nothing.	15	May optionally not transmit anything.	UnReset	11
Idle	X	0	0	Set EFCB = 1.	0	Ack.	Idle	12
	X	1		Do nothing.	15	May optionally not transmit anything.	Idle	13
	X	X	1,5..8, 10..15	Do nothing.	15	May optionally not transmit anything.	Idle	14
	X	0	2	Do nothing.	15	May optionally not transmit anything.	Idle	15
	== EFCB	1		Toggle EFCB	0	Ack.	Idle	16
	!=EFCB			Do nothing.	–	Re-transmit most recent response that contained function code 0 (ACK) or 1 (NACK).	Idle	17
	X	0	3	Do nothing.	15	May optionally not transmit anything.	Idle	18
	== EFCB	1		Send request to application parsing routine. Toggle EFCB.	0	Ack.	Idle	19

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Current state	Event that triggers an action and possible transition			Action			Transition to state	
A	B			C	D		E	
If the software state is	and a frame is received with link control octet fields			then perform this action	and send this reply		and go to this state	
	FCB	FCV	Func		Func	Comments		
	!=EFCB			Do nothing.	0	Ack.	Idle	20
Idle	X	0	4	Send request to application parsing routine.	Do not transmit anything		Idle	21
	X	1		Do nothing.			Idle	22
	X	0	9	Do nothing.	11	Status of link.	Idle	23
	X	1		Do nothing.	15	May optionally not transmit anything.	Idle	24

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

9.3.2.3 Secondary station state diagram

Figure 63—Secondary station state diagram

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

9.4 Sample CRC calculations

The following are 'C' language coding examples that illustrate how to compute the DNP3 CRC.

```
/*********************************************************************
***
This function illustrates how to compute the CRC and append it to a data
block for transmission.
*********************************************************************
**/

short idx;                    // Index
unsigned char dataBlock[18];  // Array to hold transmitted block
short blockSize;              // Size of data block, not including CRC
octets
unsigned short crc;           // 16-bit check code (crc accumulator)

// ...
// Insert code here to load contents of dataBlock and compute blockSize;
// ...

// Compute check code
crc = 0; // Initialize
for (idx = 0; idx < blockSize; idx++)
  computeCRC(dataBlock[idx],&crc);
crc = ~crc; // Invert

// Append CRC to end of block

dataBlock[idx++] = (unsigned char)crc;
dataBlock[idx] = (unsigned char)(crc >> 8);

// ...
```

[Code Continued Next Page]

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

```
/**********************************************************************
***
This function illustrates how to compute the CRC and check it for a
received data block.
**********************************************************************
**/

short idx;                    // Index
unsigned char dataBlock[18];  // Array to hold received data block
short blockSize;              // Size of data block, not including CRC
octets
unsigned short crc;           // 16-bit check code (crc accumulator)

// ...
// Insert code here to receive contents of dataBlock and blockSize;
// ...

// Compute check code for data in received block
crc = 0; // Initialize
for (idx = 0; idx < blockSize; idx++)
  computeCRC(dataBlock[idx],&crc);
crc = ~crc; // Invert

// Check CRC at end of block

if (dataBlock[idx++] == (unsigned char)crc &&
    dataBlock[idx] == (unsigned char)(crc >> 8))
  {
  // Block received without error

  // ...
  // Insert code here to process good block
  // ...
  }
else
  {
  // Error discovered

  // ...
  // Insert error processing code here
  // ...
  }

// ...
```

[Code Continued Next Page]

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

[Table Lookup Method]

```
/*********************************************************************
*
This function updates the contents of *crcAccum using a table lookup
method.
*********************************************************************
/
void computeCRC
  (
  unsigned char dataOctet, // Data octet
  unsigned short *crcAccum // Pointer to 16-bit crc accumulator
  )
{

static unsigned short crcLookUpTable[256] =  // Look up table values
  {
  0x0000, 0x365E, 0x6CBC, 0x5AE2, 0xD978, 0xEF26, 0xB5C4, 0x839A,
  0xFF89, 0xC9D7, 0x9335, 0xA56B, 0x26F1, 0x10AF, 0x4A4D, 0x7C13,
  0xB26B, 0x8435, 0xDED7, 0xE889, 0x6B13, 0x5D4D, 0x07AF, 0x31F1,
  0x4DE2, 0x7BBC, 0x215E, 0x1700, 0x949A, 0xA2C4, 0xF826, 0xCE78,
  0x29AF, 0x1FF1, 0x4513, 0x734D, 0xF0D7, 0xC689, 0x9C6B, 0xAA35,
  0xD626, 0xE078, 0xBA9A, 0x8CC4, 0x0F5E, 0x3900, 0x63E2, 0x55BC,
  0x9BC4, 0xAD9A, 0xF778, 0xC126, 0x42BC, 0x74E2, 0x2E00, 0x185E,
  0x644D, 0x5213, 0x08F1, 0x3EAF, 0xBD35, 0x8B6B, 0xD189, 0xE7D7,
  0x535E, 0x6500, 0x3FE2, 0x09BC, 0x8A26, 0xBC78, 0xE69A, 0xD0C4,
  0xACD7, 0x9A89, 0xC06B, 0xF635, 0x75AF, 0x43F1, 0x1913, 0x2F4D,
  0xE135, 0xD76B, 0x8D89, 0xBBD7, 0x384D, 0x0E13, 0x54F1, 0x62AF,
  0x1EBC, 0x28E2, 0x7200, 0x445E, 0xC7C4, 0xF19A, 0xAB78, 0x9D26,
  0x7AF1, 0x4CAF, 0x164D, 0x2013, 0xA389, 0x95D7, 0xCF35, 0xF96B,
  0x8578, 0xB326, 0xE9C4, 0xDF9A, 0x5C00, 0x6A5E, 0x30BC, 0x06E2,
  0xC89A, 0xFEC4, 0xA426, 0x9278, 0x11E2, 0x27BC, 0x7D5E, 0x4B00,
  0x3713, 0x014D, 0x5BAF, 0x6DF1, 0xEE6B, 0xD835, 0x82D7, 0xB489,
  0xA6BC, 0x90E2, 0xCA00, 0xFC5E, 0x7FC4, 0x499A, 0x1378, 0x2526,
  0x5935, 0x6F6B, 0x3589, 0x03D7, 0x804D, 0xB613, 0xECF1, 0xDAAF,
  0x14D7, 0x2289, 0x786B, 0x4E35, 0xCDAF, 0xFBF1, 0xA113, 0x974D,
  0xEB5E, 0xDD00, 0x87E2, 0xB1BC, 0x3226, 0x0478, 0x5E9A, 0x68C4,
  0x8F13, 0xB94D, 0xE3AF, 0xD5F1, 0x566B, 0x6035, 0x3AD7, 0x0C89,
  0x709A, 0x46C4, 0x1C26, 0x2A78, 0xA9E2, 0x9FBC, 0xC55E, 0xF300,
  0x3D78, 0x0B26, 0x51C4, 0x679A, 0xE400, 0xD25E, 0x88BC, 0xBEE2,
  0xC2F1, 0xF4AF, 0xAE4D, 0x9813, 0x1B89, 0x2DD7, 0x7735, 0x416B,
  0xF5E2, 0xC3BC, 0x995E, 0xAF00, 0x2C9A, 0x1AC4, 0x4026, 0x7678,
  0x0A6B, 0x3C35, 0x66D7, 0x5089, 0xD313, 0xE54D, 0xBFAF, 0x89F1,
  0x4789, 0x71D7, 0x2B35, 0x1D6B, 0x9EF1, 0xA8AF, 0xF24D, 0xC413,
  0xB800, 0x8E5E, 0xD4BC, 0xE2E2, 0x6178, 0x5726, 0x0DC4, 0x3B9A,
  0xDC4D, 0xEA13, 0xB0F1, 0x86AF, 0x0535, 0x336B, 0x6989, 0x5FD7,
  0x23C4, 0x159A, 0x4F78, 0x7926, 0xFABC, 0xCCE2, 0x9600, 0xA05E,
  0x6E26, 0x5878, 0x029A, 0x34C4, 0xB75E, 0x8100, 0xDBE2, 0xEDBC,
  0x91AF, 0xA7F1, 0xFD13, 0xCB4D, 0x48D7, 0x7E89, 0x246B, 0x1235
  };

*crcAccum =
     (*crcAccum >> 8) ^ crcLookUpTable[(*crcAccum ^ dataOctet) & 0xFF];
} /* end computeCRC() */
```

[Code Continued Next Page]

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

[Bit-shifting Method]

```
/**********************************************************************
*
This function updates the contents of *crcAccum using right shifts for
each bit.
**********************************************************************
/
void computeCRC
  (
  unsigned char dataOctet, // Data octet
  unsigned short *crcAccum // Pointer to 16-bit crc accumulator
  )
{
unsigned char idx;    // Index
unsigned short temp;  // Temporary variable

#define REVPOLY 0xA6BC // Reverse polynomial for right shifts

// Perform right shifts and update crc accumulator
for (idx = 0; idx < 8; idx++)
  {
  temp = (*crcAccum ^ dataOctet) & 1;
  *crcAccum >>= 1;
  dataOctet >>= 1;
  if (temp)
    *crcAccum ^= REVPOLY;
  }
} /* end computeCRC() */
```


Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

10. Layer-independent topics

10.1 Purpose of layer-independent topics

The purpose of this clause is to elaborate on a number of topics that apply to more than a single layer.

10.2 Confirmation and retry guidelines

10.2.1 Recommendations

a) In general, disable Data Link Layer confirmations.

Note that even though a device does not request data link confirmations, it shall respond with data link confirmations when requested by another device.

b) Implementers should provide configurable Application Layer fragment sizes. This enables the user to set larger fragment sizes in low noise environments and smaller fragments in high noise environments.

c) Disable Application Layer retries in systems where the master polls frequently.

10.2.2 Background

10.2.2.1 Data Link Layer confirms and Application Layer confirms

DNP3 has two types of confirmations, Data Link Layer and Application Layer.

A Data Link Layer confirm applies when a primary-to-secondary Data Link Layer frame is transmitted with function code CONFIRMED_USER_DATA. The secondary is obligated to return a function code ACK frame as confirmation (assuming there is no reason not to). The ***ack*** response only confirms receipt of a single Data Link Layer frame.

An Application Layer Confirm applies to an entire Application Layer fragment, regardless of how many Data Link Layer frames are required to transport the whole fragment. The outstation requests Application Layer confirmation of a fragment by setting the CON bit in the Application Layer control octet. Whenever the master receives a fragment having this bit set, it is obligated to return an Application Layer confirm message having function code CONFIRM.

10.2.2.2 Why confirmations are necessary

Certain operations performed by outstation devices require a mechanism to verify that the master correctly received responses.

A classic example is the processing of binary input data changes queued in the outstation device. The outstation shall verify that the data transmitted to the master remains in the proper chronological order so that the end user is able to re-create the real-world sequence of status changes in the outstation. The mechanism shall verify that no data is lost in the transmission process.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Continuing with the preceding example, an outstation device may have 100 binary changes queued to report to its master station. However, the master station may request a maximum of only 20 in the response. The outstation uses confirmations for determining whether the master received the 20 changes so that it can

— Remove those changes from its change queue.

— Know what to return in the next request.

Another use for confirmation is to inform the outstation that its unsolicited Application Layer fragments reached the master. Confirmation also assures that each Application Layer fragment in a multi-fragment response arrives at the master before permitting transmission of the next fragment.

10.2.2.3 Retransmissions

When a DNP3 device requests either an Application Layer or a Data Link Layer confirmation, it expects the confirmation to arrive within a configurable time. If the confirmation is not received within this period, the request is declared "timed-out," and the device can either give up for this transmission or try again, re-transmitting the exact same message. The number of times that devices retry is often configured to a relatively low number, such as 3.

Retransmissions can cause synchronization errors in the dialog between an outstation device and a master if one party retransmits when the other was not expecting it. In particular, this can cause collisions in an environment where polling is frequent.

Retries can be used in some cases, but require careful consideration

— In half-duplex systems, it is possible to configure polling intervals and timeouts long enough to allow for all possible retries from a device. However, this practice sometimes results in unacceptably long polling intervals.

— The use of collision avoidance in the Data Link and/or physical layers makes retries possible.

— Retries are required in the case of unsolicited responses. Even though collision avoidance is necessary as a matter of course, there is the possibility of losing a master confirmation message due to noise or some other cause. Outstations shall require confirmation that the master station received its responses.

The successful application of retries using collision avoidance techniques depends on several factors. Among the most important are the robustness of the collision avoidance mechanism employed and the load on the communications channel. If the mechanism is unable to avoid a collision, retries may aggravate the situation by causing more collisions and thus significantly reduce the effective bandwidth of the communications channel.

As with any communications system, the designer should pay careful attention to bandwidth allocation and management for a successful system design. In particular, overloading communications circuits has undesirable effects on system performance. In a frequently polled configuration, overloaded circuits elongate polling cycles (e.g., decrease scans per second). In an unsolicited response configuration, overloaded circuits can render response times non-deterministic during periods of peak activity.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

10.2.3 Discussion

10.2.3.1 Why Application Layer confirms are preferred

10.2.3.1.1 Data Link Layer confirms are redundant

The data returned in many outstation responses fit into a single Data Link Layer frame. This results in there being one Data Link Layer frame for each Application Layer message. Because outstations are required by the protocol rules to request an Application Layer confirmation for critical data, a Data Link Layer confirmation is redundant.

10.2.3.1.2 Bandwidth

Assuming that every Data Link Layer frame contains the maximum 292 octets, then the 10 octets required for Data Link Layer confirmation account for only 3.3 percent of the transmission time. However, the percentage increases dramatically when latencies in the channel (such as propagation delay and request-to-send, clear-to-send timings) are included, and when the message size decreases. For example, at 9600 baud, with 32 octets of user data, and channel latency of 15 milliseconds, the confirmation message consumes 29 percent of the bandwidth.

10.2.3.1.3 Assures understanding not just reception

Application Layer confirmations verify that properly assembled messages arrive at the master and that the master's Application Layer understands the data that it received. Data Link Layer confirmations only assure that Data Link Layer frames arrive without bit errors. For critical data, arrival with no bit errors is insufficient justification to delete the information from an outstation's event queues.

10.2.3.1.4 Noisy environments

Smaller packet sizes are recommended when bit error rates are high. In high bit-error-rate (noisy) environments, the probability of one or more bit errors appearing in a packet increases as the packet size increases. Conversely, the smaller the packet size, the lower the probability of an error occurring in a packet.

The original belief was that it would be better to accept the overhead of confirming each Data Link Layer frame of a multi-frame message, and re-transmit corrupted frames, than to re-send an entire Application Layer fragment. However, there is another alternative—reduce the fragment size and use only Application Layer confirmations. When fragments are reduced to the size of a Data Link Layer frame, the overhead of Application Layer confirmations, and the probability of noise corrupting those confirmation messages, is nearly the same as for Data Link Layer confirmations.

NOTE—There is a limit to reducing the size of an Application Layer fragment. Its size should be large enough to hold entire DNP3 objects—objects may not be divided between fragments. Refer to Clause 4 through Clause 6 of this standard.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

10.2.3.2 Retries and polling

10.2.3.2.1 Media access control

Use of outstation Application Layer and Data Link Layer retries are discouraged in frequent polling environments. When an outstation device requests a confirmation from the master and does not receive it, the outstation should wait for the next master request to send the information again. The reason is that it is highly likely the master has moved on to poll the next outstation in its list, and if the remote device were to retry, there is a high probability of a collision with the master's request message or the other outstation's response.

Even in the case where there is a single outstation, the outstation device could retry while the master is sending the next request to this outstation.

In a polling environment, each master request can be viewed as an invitation for an outstation device to transmit data on the shared communications circuit. The master controls which device can transmit so that collisions do not occur, and the timing of responses is predictable under all situations. Masters can ask again if a response is not received, and this provides an opportunity for the outstation to re-send its data. Thus, a master effectively manages access to the media if it does not have to contend with outstations unexpectedly transmitting on their own.

10.2.3.2.2 Unsolicited responses

Retries are a requirement when unsolicited responses are used. Unsolicited data is by definition not expected by the master, and therefore the master does not know to request the data it is missing. The burden is therefore on the outstation to verify that data gets through.

10.2.3.3 Exceptions

Despite the recommendations in 10.2.1, there are times when Data Link Layer confirms are useful.

— Certain devices (especially older devices) are unable to receive unconfirmed Data Link Layer frames back-to-back.

— Experience has shown that communications are more dependable in some environments where the noise characteristics, timing, and other circumstances make Data Link Layer confirmations successful.

Therefore, designs should have the ability to request Data Link Layer confirmation, and they should have a configurable means of enabling and disabling this feature (i.e., choosing to send user data with CONFIRMED_USER_DATA or UNCONFIRMED_USER_DATA function codes).

10.3 Time synchronization

10.3.1 General

DNP3 outstation devices are often required to time tag events to the nearest millisecond, and some devices are required to automatically freeze accumulators or perform some activity at specific times. In order to support these time-related activities, an outstation shall have a dependable time source. It may use a local

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

source, such as a GPS receiver, or it can request time synchronization from the master using DNP3 messages.

10.3.2 Time base

DNP3 time corresponds to Universal Coordinated Time (UTC).[29]

UTC does not shift with daylight saving (summer) time and does not change depending on the local time zone. Thus, events that occur at separate geographical locations, but at the same instant, are reported with identical DNP3 time values.

UTC, and therefore DNP3 time values, are adjusted appropriately when leap-seconds are added or subtracted by the International Earth Rotation Service (IERS)[30] so that UTC agrees with astronomical time based on the rotation of the earth.

Values within DNP3 time objects contain the number of milliseconds from the DNP3 time epoch, which is defined as 1970-01-01T00:00:00.000[31] UTC. The DNP3 time epoch assumes there were exactly 86 400 000 milliseconds in every day during the intervening years; i.e., it does not include leap-seconds.

DNP3 cannot represent times that occur within the 1000 millisecond period when a leap-second is added to UTC. Devices are expected to continue counting milliseconds as usual during periods of leap-second adjustments until the outstation is re-synchronized. There may be an uncertainty in the DNP3 time measurement during this interval, and time stamps recorded in that period may be in error.

As an example of a DNP3 time computation, consider that the correctly computed DNP3 time value for 2008-01-01T00:00:00.000 UTC is 0x011732A5C400.

NOTE—DNP3 transmits time values least significant octet first as 00 C4 A5 32 17 01.

The equivalent decimal number is 1,199,145,600,000.

IERS added 23 leap-seconds between 1970 and 2008. Because of the added leap-seconds, 1 199 145 623 000 milliseconds actually elapsed between the start of 1970 and the start of 2008. However, in order to conform to the simplified rule that every day contains 86 400 000 milliseconds, the DNP3-time-to-calendar-time conversion neglects these leap-seconds. In the same way, DNP3 time shall ignore any future leap-second adjustments of UTC.

For another example, the correctly computed DNP3 time value for 1998-01-01T00:00:00.000 UTC is 0x00CDBB6D5400 (decimal 883 612 800 000). IERS added two leap-seconds between January 1998 and January 2008; however, comparison of the DNP3 time values shows a difference of 315 532 800 000 milliseconds, demonstrating that DNP3 time does not include these leap-seconds.

10.3.3 Messages for time synchronization

When an outstation anticipates that its timing reference (such as a crystal oscillator) may drift beyond the needed accuracy, it should set the IIN1.4 [NEED_TIME] bit in responses. The master shall send the time promptly after receiving a response with this bit set.

[29] The effective date for using the UTC time base is 1 January 2008. Prior to this date, DNP3 did not require a specific time reference.
[30] http://www.iers.org/.
[31] The date format is YYYY-MM-DDThh:mm:ss.sss where ss.sss represents seconds and milliseconds and where a capital letter T separates the date and time fields. This notation conforms to ISO 8601 [B12].

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

NOTE | Outstations that set the IIN1.4 [NEED_TIME] bit at unreasonably short intervals could adversely impact system operation by dedicating a disproportionate amount of processing to non-data collection activities.

10.3.3.1 Non-LAN procedure

Figure 64 shows the sequence and relative timing of messages used to time synchronize an outstation from the master. Two separate request messages are transmitted from the master, ***delay measurement*** and ***write***. The first measures the propagation delay in the physical layer and the second sends the time that it is expected to be when the message arrives at the outstation.

There are seven critical time points in the sequence. Letters inside square brackets [A] through [G] show these. These points, except [G], are the times when the leading edge of the start bit either begins transmission over the physical media or is detectable at the receiving end. This would be in the starting 0x05 octet in the Data Link Layer frame. [G] represents the time when the clock is set.

Figure 64—Timing of non-LAN time synchronization

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

The steps for time synchronization are as follows:

a) The first request message sent from the master uses function code DELAY_MEASURE. The master records the exact instant when the leading edge of the start bit of the first octet in the transmitted request message leaves the master device. This is critical time point [A] in the diagram.

b) The outstation records the time at the instant when the leading edge of the first start bit is detected. This is critical time point [B] in the diagram.

c) The outstation needs to formulate a response containing a single Fine Time Delay DNP3 object, group 52, variation 2. The Fine Time Delay object holds the outstation processing delay that is the time difference, in milliseconds, from critical time point [B] to critical time point [C]. In order to formulate such a message, the outstation shall either

   1) Anticipate when [C] will occur (using a guaranteed time of transmission or other technique).

   2) Dynamically create the response message on the fly.

d) The master shall record the exact instant when the leading edge of the start bit of the first octet in the received response message arrives at the master device. This is critical time point [D].

e) The master calculates the propagation delay time as follows:

   (Time at [D] – Time at [A] – outstation processing delay) / 2.

f) The master transmits a message with an Absolute Time and Date object, group 50, variation 1, having a value that is the time that it will be at critical time point [F]. This requires the master to add the propagation delay time computed in step e to the true time at critical time point [E]. In order to formulate such a message, the master shall either

   1) Anticipate when [E] will occur (using a guaranteed time of transmission or other technique).

   2) Dynamically create the response message on the fly.

g) The outstation shall record the exact instant when the leading edge of the start bit of the first octet in the received ***write*** request message arrives at the outstation device. This is critical time point [F].

h) The outstation sets its internal clock using the time in the ***write*** request and knowing the difference in time between critical time points [F] and [G]. Critical time point [G] is the instant when the clock is set. The outstation's time is calculated as

   Time in request + (Number of milliseconds from [F] to [G])

NOTE The designers of masters and outstations should consider that there can exist internal delays caused by the operating system and other issues. Furthermore, in systems where the clear-to-send signal from a modem controls when transmission begins, this timing can be imprecise. It is important to accommodate these variables when accurate time synchronization is mandatory.

i) The outstation sends a response.

10.3.3.2 LAN procedure

Figure 65 shows the sequence and relative timing of messages used to time synchronize an outstation from the master. Two separate request messages are transmitted from the master, ***record current time*** and ***write***. The first requests the outstation to record the time of receipt of the last octet in the message, and the second sends the time that the message should have arrived at the outstation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 65—Timing of LAN time synchronization

A different method is used for time synchronization over a LAN than is used in non-LAN applications. The reasons are that the propagation delay measurement is negligible and that recording the time at the first bit of the first octet does not account for network collisions.

NOTE It is not practical to perform time synchronization over IP routers and gateways as their delays are inconsistent and vary with network loading.

There are three critical time points in the sequence. Letters inside square brackets [A] through [C] show these. [A] represents the instant when the last message octet has left the Ethernet hardware, and [B] represents the instant when the last octet of the message is received in the Ethernet hardware. The last octet is the second octet of the last CRC value in a Data Link Layer frame. [C] represents the instant when the clock is set.

The steps for time synchronization are as follows:

a) The master sends a message having function code RECORD_CURRENT_TIME. The master records the time when the last octet leaves the Ethernet hardware. This is critical time point [A] in the diagram.

b) The outstation records the time (or starts a timer) when the last octet of the request arrives in its Ethernet hardware. This is critical time point [B] in the diagram.

c) The outstation sends a null response.

d) The master sends a message having function code WRITE with a Last Recorded Time and Date object, group 50, variation 3. The value in this object contains the time that the master recorded in step a).

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

e) The outstation sets its internal clock using the time in the ***write*** request and knowing the difference in time between critical time points [B] and [C]. Critical time point [C] is the instant when the clock is set. The outstation's time is calculated as

Time in request + (Number of milliseconds from [B] to [C])

f) The outstation sends a null response. IIN1.4 [NEED_TIME] should be cleared in this response.

A master should transmit the ***write*** request message soon after the ***record current time*** request, the closer the better; however, intervening messages are permitted.

Outstations shall be prepared to receive and process other messages, from the same master or from different masters, which arrive between the first and second time synchronization messages.

If an outstation receives another ***record current time*** message without an intervening ***write*** request message, it shall discard the original recorded time and save the newer time from the most recently received ***record current time*** message.

This procedure may use broadcast messages to synchronize multiple outstations simultaneously since it is assumed that the ***record current time*** and ***write*** messages are received by all of them at the same instant. The master uses UDP to send the broadcast datagram. Outstations do not respond to broadcast messages.

NOTE

This procedure requires that the Ethernet driver provide the ability to record the last octet time on transmission and receive. Processor speed, interrupt latencies, Internet protocol stack design, etc. greatly affect the time synchronization process. Time accuracy to within a millisecond is not possible without this capability.

If a master or outstation device cannot record time at the Ethernet driver, it should do so as close as possible to the Internet stack interface. This introduces an error factor in the synchronization process. A device may wish to minimize this error by adding a predetermined bias value that reflects an average delay through the Internet stack and Ethernet driver. This compensation factor does not take into account the delay caused by Ethernet collisions and retransmission; however, it may reduce the average error to a level that can be tolerated in the system.

10.3.4 Time synchronization retries

10.3.4.1 Requirements

Masters and outstations shall not retry time synchronization messages at either the Application or Data Link Layers. These messages apply to those having Application Layer function codes:

— DELAY_MEASURE request from master and corresponding response (RESPONSE function code) from outstation.

— RECORD_CURRENT_TIME requests from master.

— WRITE requests from master with an Absolute Time object, group 50, variation 1.

— WRITE requests from master with a Last Recorded Time object, group 50, variation 3.

Both master and outstations shall transport time synchronization messages using Data Link Layer function code UNCONFIRMED_USER_DATA.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

If a master does not receive the expected response on the first try, it shall send a new request and not retry the message. New requests have a different sequence number in the application control octet of the application header and cause the response content to be re-generated.

The following subclauses explain why these requirements are necessary.

10.3.4.2 Justification for non-LAN applications

A master sends the time via a ***write*** (function code WRITE) with object group 50, variation 1 to the outstation. The outstation records the time when the first bit of the first octet is received and returns a null response.

Here is what happens when the master sends a repeat Application Layer fragment with the same time as in the original transmission. If the outstation misses the original transmission, but receives the repeated message, the outstation will set its time incorrectly. This is because the time in the *repeated* message specifies what time the first bit of the *original* message should arrive at the outstation; it does not specify what time the first bit of the *repeated* message should arrive.

The master can erroneously compute the time delay in the communication channel when a retry happens while sending a request to measure the delay (function code DELAY_MEASURE). If the request does not arrive at the outstation on the first try but does on the second, then when the response returns from the outstation, the master would incorrectly compute the time for the round trip by using the time when it sent the first try as a reference.

In another scenario, the outstation receives both requests to measure the channel delay, but its first response does not arrive at the master. Upon receipt of the retried request, the outstation simply repeats its first response. However, this would be incorrect if the time required for the outstation to process the message varies from message to message—typical of multi-tasking software. The value in the returned Time Delay Fine object would not represent the true outstation processing time for the *retry*, consequently causing the master to compute the channel delay time incorrectly.

Based on similar logic, it follows that repeated Data Link Layer frames can also affect time computations.

10.3.4.3 Justification for LAN applications

A problem can occur when a master sends a repeat of an Application Layer function code RECORD_CURRENT_TIME message. The master records the time upon sending the original transmission. If the outstation misses the original transmission, but receives the repeated message, the outstation will calculate an erroneous new time upon receipt of the follow-up ***write*** request having a time object (group 50, variation 3). This is because the outstation bases the computation on the *repeated* message and the master bases the time on the *original*.

Another error condition can occur when the master records the current time in the original message and then re-records the current time in the repeated ***record current time*** message. If the outstation receives the first transmission but its response does not arrive at the master, and the master sends a repeat request to record the time, the outstation will not process it. Then subsequently when the master transmits the ***write*** request with the time object (group 50, variation 3), that time is with respect to the *repeated* message and not to the *original* message that the outstation used to record its time.

Again, based on similar logic, it follows that repeated Data Link Layer frames can also affect time computations.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

NOTE—Ethernet retransmissions are not considered retries because they are not a part of the DNP3 protocol. They do, however, limit the accuracy of the time synchronization scheme.

10.4 Handling multiple messages

In DNP3, reception frequently involves more than a single set of request or response octets. Devices are permitted to transmit two messages consecutively without an interval between them. This subclause provides requirements for handling multiple messages, or multiple parts of a message, and discusses how these situations can occur.

10.4.1 Requirements

— **Rule 1:** Masters shall be able to receive a minimum of one more than the total of all of the Data Link Layer frames necessary for a maximum-sized Application Layer fragment without requiring a gap between Data Link Layer frames. To evaluate this requirement, transmit the application data from the outstation using Data Link Layer function code UNCONFIRMED_USER_DATA.

— **Rule 2:** Outstations shall be able to receive an Application Layer confirm (Application Layer CONFIRM function code) immediately followed by an Application Layer request without requiring a delay between them. To evaluate this requirement, transmit the Application Layer confirm from the master using the Data Link Layer function code UNCONFIRMED_USER_DATA.

10.4.2 Back-to-Back confirmation and request/response

Back-to-back Data Link Layer frames can happen when, for example, a master sends an Application Layer request to the outstation using a Data Link Layer frame with the CONFIRMED_USER_DATA function code. The outstation shall return a Data Link confirm frame followed immediately by a Data Link Layer frame containing the Application Layer response data. In outstations with fast processors, it is possible for there to be no gap occurring between the transmitted frames.

Back-to-back Application Layer fragments can happen when an outstation sends event data. The outstation shall request the master to return Application Layer verification that the message arrived. The master may transmit the confirmation (using the CONFIRM Application Layer function code) immediately followed by another request—both messages being transmitted using the Data Link Layer UNCONFIRMED_USER_DATA function code.

Thus, both masters and outstation devices shall be prepared to receive back-to-back messages without any time gaps separating messages.

10.4.3 Back-to-Back without confirmation

Outstations can send Application Layer fragments that require more than one Data Link Layer frame. A full Application Layer fragment having a size of 2048 octets requires nine frames, assuming maximum length frames. DNP3 does not specify the time between frames; therefore, masters shall be prepared to accept multiple Data Link Layer frames, one-after-the-other, when the outstation does not request Data Link confirmation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

10.4.4 Multi-drop communications

In some communication systems, outstations are able to receive octets from the master station and the responses from other outstations. Thus, with fast processor devices in the system, the master station request messages and outstation response messages may abut each other with little or no gaps between them. Although there is not a specific DNP3 requirement for this situation, outstations should be prepared to receive multiple Data Link Layer frames, without intervening gaps, addressed to itself and to other devices.

10.4.5 Unsolicited responses

Masters that support unsolicited messages shall be prepared to accept messages at random times from multiple outstations. Furthermore, masters shall anticipate that an unsolicited response can arrive from an outstation while the master is waiting for a solicited response to a request it sent to another outstation. Software shall be designed to properly process Application Layer fragments, Transport Function segments, and Data Link Layer frames as they arrive.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

11. Data object library

11.1 Data object library—part one

11.1.1 Overview

DNP3 uses objects to communicate values and information between devices. An analog input point, for example, has a value and may have other attributes such as whether that value exceeds the measurement range. In order to transport the value and attributes, a standardized way to represent this information within a message exists in DNP3 so that receivers know how to interpret the message contents. DNP3 uses group numbers to categorize the data type, and generally, it uses variation numbers to specify how the data from within the group is encoded.[32] Each instance of an encoded information element, defined by a unique group and variation within the message, is called a DNP3 object. Point index numbers differentiate multiple instances of DNP3 objects of the same type.

The DNP3 Object Library contains a description and encoding for all of the DNP3 objects. The library does not specify anything about the physical devices that are represented by these objects. Subclause 11.1.9 provides descriptions of various point types and guidance for choosing objects. A DNP3 vendor is generally free to select which DNP3 object is used to represent each of the tangible items. Nevertheless, there are guidelines specified in the standard DNP3 implementation levels and industry norms that are outside the scope of this document.

11.1.2 Library documentation organization

The DNP3 Object Library documentation is organized into two sections. This clause contains general information and the rules needed to interpret the objects. It also contains definitions and descriptions that are common to all or many of the objects.

This clause defines the DNP3 objects. Each object group and variation is described in an individual insert consisting of one or more pages. Inserts facilitate document maintenance when the DNP3 organization adds, updates, or deletes groups and variations.

11.1.3 Primitive data types

11.1.3.1 Summary of types

The formal definition of object group and variations depends on primitive data types that are commonly known to software developers. A list of the types and a brief description appear in Table 52. Subsequent paragraphs describe the types in detail.

[32] There are some situations where the variation specifies more information than just the data format.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 52—Primitive data types

Basic data type	Brief description
BSTRn	n-**bit** bit string.
UINTn	n-**bit** unsigned integer.
INTn	n-**bit** signed integer.
FLTn	n-**bit** floating-point, formatted per IEEE Std 754™.
BCDn	n-**BCD character** binary-coded decimal.
VSTRn	n-**octet** printable ASCII text string, **not** null-terminated.
OSTRn	n-**octet** octet string.
SET of n	n **sets** where each set is similarly constructed of a sequence of other basic data types.
VARIANT	The basic data type depends on the specific object instance.
UNCDn	n-**octet** Unicode UTF-8 string, **not** null-terminated.

11.1.3.2 Numeric notation conventions

This subclause discusses how various numeric values are expressed in the library.

11.1.3.2.1 Decimal values

Decimal values use the digits 0 through 9 and may include a sign. Examples are 20, –245, +7, and 12345.

11.1.3.2.2 Binary-code decimal values

Binary-coded decimal values use the digits 0 through 9 and may have a sign. The suffix, bcd, is used to explicitly identify a value as being the BCD type. Examples are 0341bcd, –01991bcd, and 999999bcd.

11.1.3.2.3 Hexadecimal values

Hexadecimal values use the digits 0 through 9, and the letters A through F, in either uppercase or lowercase. All hexadecimal values are preceded with the character pair 0x. Examples are 0x3C, 0xF7A5, and 0x5D40F790.

11.1.3.2.4 Binary values

Binary values use the digits 0 and 1 and have the suffix letter b. Examples are 0110b, 0b, and 1100b.

11.1.3.2.5 Floating-point values

Floating-point values are expressed in one of two ways. The first uses the digits 0 through 9, a decimal point, and may include a sign. Examples are 25.6, –195.23, 0.0, and +7841.01. The decimal point is mandatory.

The second method uses scientific notation and takes the form of a mantissa followed by an exponent stated as a power of ten. The mantissa uses the digits 0 through 9, a decimal point, and may include a sign. The decimal point is mandatory. The exponent begins with the letter E, in either uppercase or lowercase. An optional sign follows the E and the digits 0 through 9. Examples are –25.11E–2 and 1.037625148E12.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

11.1.3.3 Bit strings

Bit string fields or values use the notation BSTRn, where n represents the number of bits in the string. For example, BSTR7 has seven bits. Bit strings are expressible with binary values such as 1100110b. The right-most bit of the string corresponds to the least significant bit of the binary value—zero in the example given in the preceding sentence.

When bit strings appear in an octet, the least significant bit of the string occupies the lowest bit position within the field. Here are two examples, one where the field begins in the octet's bit 0 and the second where the field begins in the octet's bit 3. A BSTR4 having a value of 1100b is illustrated in the shaded bit positions.

7	6	5	4	3	2	1	0	← bit position
				1	1	0	0	

7	6	5	4	3	2	1	0	← bit position
	1	1	0	0				

Long bit strings wrap into subsequent octets beginning in the lower bit positions. The next illustration shows a BSTR14 having a value of 10111100001010b in the shaded bit positions. Two octets are required.

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
0	0	0	0	1	0	1	0	
		1	0	1	1	1	1	

Another example shows a BSTR18 having a value of 110011111100001010b in the shaded bit positions. The field begins in bit 2 of the first octet and continues into the third octet.

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
0	0	1	0	1	0			
1	1	1	1	1	1	0	0	
				1	1	0	0	

11.1.3.4 Unsigned and signed integers

Unsigned integer fields or values use the notation UINTn, where n represents the number of bits in the integer. For example, UINT32 denotes a 32-bit unsigned integer. Signed integer fields or values use the notation INTn, where n represents the number of bits in the integer including the sign bit, which appears in bit position n – 1. For example, INT16 specifies a 16-bit signed integer.

11.1.3.4.1 Characteristics

— Unsigned integers always have a positive value that ranges from 0 to $2^n - 1$.

— Signed integers are expressed as a twos-complement value that ranges from $-2^{(n-1)}$ to $2^{(n-1)} - 1$.

— Unless otherwise specified on an insert page, integers require an integral number of 8 bits. Thus, typically the objects state them as INT16, INT32, UINT16, UINT32, and UINT48.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— Unless otherwise specified on an insert page, integers are transmitted little-endian style; that is, the least significant octets are transmitted first, and the most significant octets are transmitted last.

— Unless otherwise specified on an insert page, the least significant bit of an integer occupies the bit 0 position in an octet, and the most significant bit of an integer occupies the bit 7 position in an octet.

11.1.3.4.2 Example

This is an example of a 16-bit signed integer having a value of –22 000 followed by a 32-bit unsigned integer having a value of 4 000 000 000.

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	0	1	0	0	0	0	b0	INT16 value = –22 000
b15	1	0	1	0	1	0	1	0		
	0	0	0	0	0	0	0	0	b0	
	0	0	1	0	1	0	0	0		UINT32 value = 4 000 000 000
	0	1	1	0	1	0	1	1		
b31	1	1	1	0	1	1	1	0		

Notice that the first and last bit of each value is shown in the preceding figure. b0 is the least significant bit, and b15 and b31 are the most significant bits.

11.1.3.5 Floating-point values

Floating-point values use the notation FLTn, where n represents the number of bits in the value. For example, FLT32 denotes a 32-bit floating-point value. DNP3 uses two floating-point formats, and each type has a name.

— FLT32 format is called float or single-precision or short floating-point.

— FLT64 format is called double-precision.

> NOTE—Previous DNP3 documentation referred to FLT64 as a “long floating-point.” This has changed in version 2.00 to conform to common programming practices.

11.1.3.5.1 Characteristics

Floating-point numbers are constructed in conformance with IEEE Std 754. The basic format is

<sign>**1.**<mantissa>*2$^{\text{<biased exponent>}}$

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

where the sign, mantissa, and biased exponent values are obtained from specific bit fields as shown:

— FLT32 values are formatted as

31	30 23	22 0	← bit position
Sign	Biased Exponent	Mantissa	

A sign bit of 0 represents a positive number, and a 1 in the sign bit position specifies a negative value.

The mantissa is a 23-bit value. There is an implied 1 and a binary point to the left of bit 22.

The exponent is a power of two. The true exponent is obtained by subtracting 127 from the biased exponent. The legal ranges of the biased exponent are 1 to 254, corresponding to true exponents of –126 to +127. This leaves two special case biased exponent values of 0 and 255.

When the biased exponent is 255 and the mantissa is all zeros, this represents + or – infinity depending on the sign bit. When the biased exponent is 255 and the mantissa is not zero, then this represents NAN, not-a-number.

When the biased exponent is 0, the mantissa represents a de-normalized number. The implied 1 in the 24th bit disappears and the value is less than 2^{-126}.

— FLT64 values are formatted as

63	62 52	51 0	← bit position
Sign	Biased Exponent	Mantissa	

A sign bit of 0 represents a positive number, and a 1 in the sign bit position specifies a negative value.

The mantissa is a 52-bit value. There is an implied 1 and a binary point to the left of bit 51.

The exponent is a power of two. The true exponent is obtained by subtracting 1023 from the biased exponent. The legal ranges of the biased exponent are 1 to 2046, corresponding to true exponents of –1022 to +1023. This leaves two special case biased exponent values of 0 and 2047.

When the biased exponent is 2047 and the mantissa is all zeros, this represents + or – infinity depending on the sign bit. When the biased exponent is 2047 and the mantissa is not zero, then this represents NAN, not-a-number.

When the biased exponent is 0, the mantissa represents a de-normalized number. The implied 1 in the 53rd bit disappears and the value is less than 2^{-1022}.

— FLT32 values are capable of expressing numbers between 0 and $\pm 3.4 \times 10^{38}$.

— FLT64 values are capable of expressing numbers between 0 and $\pm 1.7 \times 10^{308}$.

— Unless otherwise specified on an insert page, floating-point values are transmitted little-endian style; that is, the least significant octets are transmitted first, and the most significant octets are transmitted last.

— Unless otherwise specified on an insert page, the least significant bit of floating-point value occupies the bit 0 position in an octet, and the most significant bit of a floating-point value occupies the bit 7 position in an octet.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

11.1.3.5.2 Example

This is an example of a FLT64, double-precision, floating-point value having a value of –2.3258890344e+03.

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position
	0	0	1	0	0	0	0	0	b0
	0	1	1	1	0	1	0	0	
	1	0	0	0	0	1	0	1	
	0	0	1	0	1	1	1	1	
	1	1	0	0	0	1	1	1	
	0	0	1	0	1	0	1	1	
	1	0	1	0	0	0	1	0	
b63	1	1	0	0	0	0	0	0	

FLT64 value =
–2.3258890344e+03

Notice that the first and last bit of the value is shown in the preceding figure. b0 is the least significant bit, and b63 is the most significant bit. The exponent bit positions are shaded for visual reference.

11.1.3.6 Binary-coded decimal

Binary-coded decimal values use the notation BCDn, where n represents the number of BCD characters. For example, BCD8 requires 8 BCD characters.

11.1.3.6.1 Coding practice

Each BCD character requires 4 bits and is coded as in Table 53.

Table 53—BCD character coding

3	2	1	0	← bit position within BCD character	
0	0	0	0	0	Equivalent character value
0	0	0	1	1	
0	0	1	0	2	
0	0	1	1	3	
0	1	0	0	4	
0	1	0	1	5	
0	1	1	0	6	
0	1	1	1	7	
1	0	0	0	8	
1	0	0	1	9	
1	0	1	0	–	In the most significant BCD character, any of these bit combinations indicates a negative value. In all other BCD character positions, these bit combinations are illegal.
1	0	1	1	–	
1	1	0	0	–	
1	1	0	1	–	
1	1	1	0	–	
1	1	1	1	–	

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

11.1.3.6.2 Characteristics

— The value of a positive BCD number is $\sum_{i=1}^{n}(\text{BCD_character_value})(10^{i-1})$, where i is the BCD digit number (i = 1 for the least significant BCD digit).

— Binary-coded decimal values range from $-10^{(n-1)}$ to $10^{n} - 1$. Negative values require at least two BCD characters.

— Typically, the objects use values that fit into 2, 4, or 8 octets and use the types BCD4, BCD8, or BCD16. Other octet lengths are permitted.

— If it is necessary to fit an odd number of BCD characters into an integral number of octets, the BCD number shall be padded with a 0 digit in the most significant digit position. For example, 173bcd is transmitted as 0173bcd, and –57bcd is transmitted as –057bcd.

— Unless otherwise specified on an insert page, BCD characters are transmitted little-endian style; that is, the least significant BCD characters are transmitted first, and the most significant BCD characters are transmitted last.

— Unless otherwise specified on an insert page, the least significant bit of a BCD value occupies either bit position 0 or 4 in an octet.

11.1.3.7 Printable ASCII strings

Printable ASCII strings are non-null-terminated strings whose character set is limited to characters that display textual characters or perform common printer control commands. Generally, these strings are meant for humans to read or write. They are typically used for names, instructions, errors reports, logged information, display notices, user identifiers, and passwords.

Printable ASCII strings use the notation VSTRn, where n represents the number of octets. The V in VSTR comes from the word “visible” and is intended to represent characters that are visible on a printer or display device.

11.1.3.7.1 Printable characters

Table 54 shows printable characters in the non-shaded cells. Only characters with values greater than 31 and less than 127 are guaranteed to be interoperable.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 54—Preferred printable characters

Character value Dec	Character value Hex	Printable symbol	Description
0	00		Null
1	01		SOH (Start of heading)
2	02		STX (Start of text)
3	03		ETX (End of text)
4	04		EOT (End of transmission)
5	05		ENQ (Enquiry)
6	06		ACK (Acknowledge)
7	07		BEL (Bell)
8	08		Backspace
9	09		Horizontal tab
10	0A		Line feed
11	0B		Vertical tab
12	0C		Form feed
13	0D		Carriage return
14	0E		SO (Shift out)
15	0F		SI (Shift in)
16	10		DLE (Data link escape)
17	11		DC1 (Device control 1)
18	12		DC2 (Device control 2)
19	13		DC3 (Device control 3)
20	14		DC4 (Device control 4)
21	15		NAK (Negative acknowledgment)
22	16		SYN (Synchronous idle)
23	17		ETB (End of transmission block)
24	18		CAN (Cancel)
25	19		EM (End of medium)
26	1A		SUB (Substitute)
27	1B		ESC (Escape)
28	1C		FS (File separator)
29	1D		GS (Group separator)
30	1E		RS (Record separator)
31	1F		US (Unit separator)
32	20		Space
33	21	!	Exclamation mark
34	22	"	Quotation mark
35	23	#	Number sign
36	24	$	Dollar sign
37	25	%	Percent sign
38	26	&	Ampersand

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Character value		Printable	Description
Dec	Hex	symbol	
39	27	'	Apostrophe
40	28	(	Left parenthesis
41	29	)	Right parenthesis
42	2A	*	Asterisk
43	2B	+	Plus sign
44	2C	,	Comma
45	2D	-	Hyphen
46	2E	.	Period
47	2F	/	Slash
48	30	0	0
49	31	1	1
50	32	2	2
51	33	3	3
52	34	4	4
53	35	5	5
54	36	6	6
55	37	7	7
56	38	8	8
57	39	9	9
58	3A	:	Colon
59	3B	;	Semicolon
60	3C	<	Less than
61	3D	=	Equals sign
62	3E	>	Greater than
63	3F	?	Question mark
64	40	@	Commercial at
65	41	A	A
66	42	B	B
67	43	C	C
68	44	D	D
69	45	E	E
70	46	F	F
71	47	G	G
72	48	H	H
73	49	I	I
74	4A	J	J
75	4B	K	K
76	4C	L	L
77	4D	M	M
78	4E	N	N

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Character value		Printable	Description
Dec	Hex	symbol	
79	4F	O	O
80	50	P	P
81	51	Q	Q
82	52	R	R
83	53	S	S
84	54	T	T
85	55	U	U
86	56	V	V
87	57	W	W
88	58	X	X
89	59	Y	Y
90	5A	Z	Z
91	5B	[	Left bracket
92	5C	\	Backslash
93	5D	]	Right bracket
94	5E	^	Caret
95	5F	_	Underbar
96	60	`	Grave accent
97	61	a	A
98	62	b	B
99	63	c	C
100	64	d	D
101	65	e	E
102	66	f	F
103	67	g	G
104	68	h	H
105	69	i	I
106	6A	j	J
107	6B	k	K
108	6C	l	L
109	6D	m	M
110	6E	n	N
111	6F	o	O
112	70	p	P
113	71	q	Q
114	72	r	R
115	73	s	S
116	74	t	T
117	75	u	U
118	76	v	V

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Character value		Printable	Description
Dec	Hex	symbol	
119	77	w	W
120	78	x	X
121	79	y	Y
122	7A	z	Z
123	7B	{	Left brace
124	7C	|	Vertical bar
125	7D	}	Right brace
126	7E	~	Tilde
127	7F		DEL
128	80	€	Euro
129	81		
130	82	‚	Comma
131	83	*ƒ*	Script f
132	84	„	Down double quote
133	85	…	Dots
134	86	†	Dagger
135	87	‡	Double dagger
136	88	ˆ	Circumflex
137	89	‰	
138	8A	Š	
139	8B	‹	Left caret
140	8C	Œ	OE ligature
141	8D		
142	8E	Ž	
143	8F		
144	90		
145	91	‘	Left quote
146	92	’	Right quote
147	93	“	
148	94	”	
149	95	•	Big center dot
150	96	–	Dash
151	97	—	Double dash
152	98	˜	Tilde
153	99	™	Trade mark
154	9A	š	
155	9B	›	Right caret
156	9C	œ	oe dipthong
157	9D		
158	9E	ž	

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Character value Dec	Hex	Printable symbol	Description
159	9F	Ÿ	Y umlaut
160	A0		Non-breaking space
161	A1	¡	Inverted exclamation
162	A2	¢	Cent sign
163	A3	£	Pound sterling
164	A4	¤	Currency sign
165	A5	¥	Yen sign
166	A6	¦	Broken vertical bar
167	A7	§	Section sign
168	A8	¨	Umlaut
169	A9	©	Copyright
170	AA	ª	Feminine ordinal
171	AB	«	Left angle quote
172	AC	¬	Not sign
173	AD	-	Soft hyphen
174	AE	®	Registered trademark
175	AF	¯	Macron
176	B0	°	Degree sign
177	B1	±	Plus or minus
178	B2	²	Superscript two
179	B3	³	Superscript three
180	B4	´	Acute accent
181	B5	µ	Micro sign
182	B6	¶	Paragraph sign
183	B7	·	Middle dot
184	B8	¸	Cedilla
185	B9	¹	Superscript one
186	BA	º	Masculine ordinal
187	BB	»	Right angle quote
188	BC	¼	One-fourth
189	BD	½	One-half
190	BE	¾	three-fourths
191	BF	¿	Inverted question
192	C0	À	A grave
193	C1	Á	A acute
194	C2	Â	A circumflex
195	C3	Ã	A tilde
196	C4	Ä	A umlaut
197	C5	Å	A ring
198	C6	Æ	AE ligature

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Character value		Printable	Description
Dec	Hex	symbol	
199	C7	Ç	C cedilla
200	C8	È	E grave
201	C9	É	E acute
202	CA	Ê	E circumflex
203	CB	Ë	E umlaut
204	CC	Ì	I grave
205	CD	Í	I acute
206	CE	Î	I circumflex
207	CF	Ï	I umlaut
208	D0	Ð	Eth
209	D1	Ñ	N tilde
210	D2	Ò	O grave
211	D3	Ó	O acute
212	D4	Ô	O circumflex
213	D5	Õ	O tilde
214	D6	Ö	O umlaut
215	D7	×	Multiply sign
216	D8	Ø	O slash
217	D9	Ù	U grave
218	DA	Ú	U acute
219	DB	Û	U circumflex
220	DC	Ü	U umlaut
221	DD	Ý	Y acute
222	DE	Þ	Uppercase thorn
223	DF	ß	sz ligature
224	E0	à	a grave
225	E1	á	a acute
226	E2	â	a circumflex
227	E3	ã	a tilde
228	E4	ä	a umlaut
229	E5	å	a ring
230	E6	æ	Ae ligature
231	E7	ç	c cedilla
232	E8	è	e grave
233	E9	é	e acute
234	EA	ê	e circumflex
235	EB	ë	e umlaut
236	EC	ì	i grave
237	ED	í	i acute
238	EE	î	i circumflex

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Character value		Printable	Description
Dec	Hex	symbol	
239	EF	ï	i umlaut
240	F0	ð	Eth
241	F1	ñ	n tilde
242	F2	ò	o grave
243	F3	ó	o acute
244	F4	ô	o circumflex
245	F5	õ	o tilde
246	F6	ö	o umlaut
247	F7	÷	Division sign
248	F8	ø	o slash
249	F9	ù	u grave
250	FA	ú	u acute
251	FB	û	u circumflex
252	FC	ü	u umlaut
253	FD	ý	y acute
254	FE	þ	Lowercase thorn
255	FF	ÿ	y umlaut

11.1.3.7.2 Characteristics

— Each character occupies a single octet.

— The order for printing or display of the characters in the string is from the first octet transmitted to the last octet transmitted.

11.1.3.8 Octet string

Octet strings use the notation OSTRn, where n represents the number of octets. These strings contain coded information whose meaning depends on the context where they are used. One way of looking at an octet string is to consider it as an array of binary-coded 8-bit elements.

11.1.3.8.1 Characteristics

— Unless otherwise specified on an insert page, the first (lowest numbered) element in the array, is transmitted first.

— Unless otherwise specified on an insert page, the least significant bit of an octet string octet occupies the bit 0 position in an octet, and the most significant bit of an octet string octet occupies the bit 7 position in an octet.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

An example of an octet string having 3 elements { 0x0A, 0x3F, 0xC7 } would appear as shown:

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
0	0	0	0	1	0	1	0	
0	0	1	1	1	1	1	1	
1	1	0	0	0	1	1	1	

11.1.3.9 SET of n

SET of n is a notation for indicating repetitive, similarly constructed sets consisting of other basic data types.

For an example, consider a hypothetical object description that contains multiple point index descriptions.

SET of n: Point index elements

{

UINT8: Length.

Specifies the number of octets in the *point type* and *point index* fields.

UINT8: Point type.

Specifies the point type with which the index is associated. Point type values are the group number used to convey static data.

UINTn: Point index.

An index number relative to the point indexes used to convey data values for the specified point type in traditional DNP3 read or write requests.

}

Opening and closing curly brackets { } are used to enclose the elements within a set. Each curly bracket appears on a line by itself, as shown.

The "n" in SET of n may have a specific value, such as 8, if the number of sets is known in advance; the description would then specify SET of 8. Or descriptions can use the "n" to represent a variable number of sets where 'n' depends on some other factor.

11.1.3.10 Variant

The variant data type is used to denote that the actual data type cannot be specified without knowing the context or values that appear in other members of an object.

An example of this data type appears in the description of data set prototypes and descriptors, object group 85, variation 1 and object group 86, variation 1. Both objects have a *descriptor code field* and an *ancillary value* field. The *ancillary value field* contains UINT, OST,R or VSTR data depending on what value appears in the *descriptor code field*.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

11.1.3.11 Unicode string

Unicode strings are non-null-terminated octet sequences that are based on the UTF-8 Unicode Transformation Format of ISO/IEC 10646, which encodes each Unicode character in from 1 to 6 octets.

Unicode strings shall not prepend the Unicode character U+FEFF (also know as a zero-width, no break space) for use as a signature to hint that the string contains Unicode characters and uses UTF-8 encoding. DNP3 always specifies the data types, and this practice is superfluous.

For more information including the encoding schema, the reader is referred to IETF RFC 3629,[33] ISO/IEC 10646, and the Unicode website.[34]

11.1.3.11.1 Example encoding

The following examples are copied from IETF RFC 3629. The hexadecimal octets shown in the examples are transmitted in left-to-right order.

The character sequence U+0041 U+2262 U+0391 U+002E "A<NOT IDENTICAL TO><ALPHA>." is encoded in UTF-8 as follows:

0x41	0xE2	0x89	0xA2	0xCE	0x91	0x2E

The character sequence U+D55C U+AD6D U+C5B4 (Korean ”hangugeo,” meaning “the Korean language”) is encoded in UTF-8 as follows:

0xED	0x95	0x9C	0xEA	0xB5	0xAD	0xEC	0x96	0xB4

The character sequence U+65E5 U+672C U+8A9E (Japanese “nihongo,” meaning “the Japanese language”) is encoded in UTF-8 as follows:

0xE6	0x97	0xA5	0xE6	0x9C	0xAC	0xE8	0xAA	0x9E

The character U+233B4 (a Chinese character meaning “stump of tree”) is encoded in UTF-8 as follows:

0xE0	0xA3	8E	B4

11.1.4 Data type codes

Some of the DNP3 objects require a number (code) to indicate a type of data but without specifying exactly how many bits or octets are needed. The data type may reference one of the primitives from Annex D or a derived type, such as time, which is formatted as a UINT48. Data type codes are included in this document to help assure that objects designed in the future assign the same code numbers. Table 55 lists the codes.

[33] Annex D contains a notice of copyright that permits using this material.
[34] http://www.unicode.org.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 55—Data type codes

Code	Code name	Description
	NONE	Data type does not exist or a placeholder.
	VSTR	Visible ASCII characters suitable for print and display.
	UINT	Unsigned integer.
	INT	Signed integer.
	FLT	Floating-point.
	OSTR	Octet string.
	BSTR	Bit string.
	TIME	DNP3 time (in the form of an UINT48).
	UNCD	Unicode strings (requires up to 6 octets per character).
254	U8BS8LIST	List of UINT8–BSTR8 pairs.
255	U8BS8EXLIST	Extended list of UINT8–BSTR8 pairs. Object length is 256 plus the value in the object's length octet.

Regarding the data type codes:

— Data type codes do not specify the number of octets required to convey the value. That information is obtained elsewhere.

— Floating-point types are limited to 4 and 8 octets for transporting 32-bit and 64-bit floating-point values. No other sizes are acceptable. Implementers that use 64-bit floating-point values shall carefully consider the consequences of possible non-interoperability.

— Implementers that use Unicode shall carefully consider the consequences of possible non-interoperability in that this data type is not universally supported.

11.1.5 DNP3 object types

DNP3 objects are broadly categorized into types depending on whether the object contains a current value, contains an event value, contains control-related command data, or is used as information.

11.1.5.1 Static type

Static data is non-event data that refers to the present value of points located in an outstation device. For example, single-bit binary input data, object group 1, is of static type because it is a representation of the most recently measured, obtained, or calculated value from a physical or virtual two-state point.

11.1.5.2 Event type

Event refers to DNP3 objects that convey change information or values that result from "something of interest" having happened. For example, a single-bit binary input event, object group 2, is of event type because the value represents a change of state or quality that has occurred on a single-bit binary input point.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

11.1.5.3 Command (Cmnd) type

Command type objects are included in requests and responses that relate to controlling the state or value of an output.

11.1.5.4 Information (Info) type

Information refers to DNP3 objects that complete the understanding of a request or response. For example, absolute time, object group 50, variation 1, contains the information required to set time into an outstation.

11.1.5.5 Attribute (Attrib) type

Attribute refers to objects used to convey attributes about data or devices. The values in objects of this type do not represent real-time data for an operating system.

11.1.6 Object flags

Many of the readable data objects have variations that include flags consisting of bit string (BSTRn) fields to indicate conditions or attributes of the associated data value.

11.1.6.1 Flag definitions

The purposes and meanings of DNP3 Object Flags are defined as follows.

In the following descriptions:

— An “originating device” is one that gathers field data directly (for inputs) or issues controls directly to the field (for outputs).

— A “non-originating device” is one that obtains input data or issues control commands via a communications link.

— A “reporting device” is a device that acts as a DNP3 outstation, sending DNP3 messages to an upstream device.

Data from an originating device may arrive at the master via one or more “data concentrator” devices. In this case each device in the communications chain, other than the master, is a reporting device. This identification of various devices is illustrated in Figure 66. The terms “upstream” and “downstream” that indicate relative device hierarchy are also shown in this diagram.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 66—Identification of non-originating devices (data concentrators)

Each flag bit is described in Table 56. The ONLINE, RESTART, COMM_LOST, REMOTE_FORCED, and LOCAL_FORCED flags are common to all object group types that contain flags. The other flags are specific to particular object groups as identified in the table.

Non-originating devices always pass flags through unchanged unless otherwise indicated in the following discussion.

Each flag is "set" (has the value 1) when active and is "clear" (has the value 0) when inactive.

Table 56—Flag descriptions

Name	Functional description
ONLINE	For input data objects: If clear, the point is inactive or disabled (for example: powered-down, faulty, etc.) and unable to obtain field data. The flag may optionally be cleared by a non-originating device if communications to the originating device fail. In this case the COMM_LOST flag shall also be set. For output status objects: If clear, the output point is inactive, unavailable, out-of-service, not installed, or operating in local mode. The point may not be observable or may be not controllable. Commands sent to the point may fail. When an output point is in local mode, it shall clear this flag.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Name	Functional description
RESTART	The RESTART flag indicates that the data has not been updated from the field since device reset. Originating devices shall set this bit immediately upon restarting and keep the bit set until they have an updated value in their database. Non-originating devices shall set this bit immediately upon restart and keep the bit set until it is overwritten by collecting data from a reporting device. For input data objects: If set, the object is in the initialization state, having a value that has never been updated from the field since restart. The bit is cleared when the object is first updated. In an originating device, this is when the field value is first acquired. In a non-originating device, the bit remains set until it is overwritten by collecting data from a reporting device and that data does not have the RESTART flag set. For output status objects: The RESTART flag shall only be set while a device is restarting. In an originating device, the flag shall be cleared after the device is available to accept commands, irrespective of whether or not an output value (control) has been sent to the output object. In a non-originating device, the bit remains set until it is overwritten by collecting output status information from a reporting device and that data does not have the RESTART flag set.
COMM_LOST	COMM_LOST indicates that there is a communication failure in the path between the device where the data originates and the reporting device. This flag indicates that the value reported for the object may be stale. If set, the data value reported shall be the last value available from the originating device before communications were lost. An originating device never sets this flag. A non-originating device sets this flag if it loses communication with the adjacent downstream device; otherwise it propagates the state of COMM_LOST flag as received from the downstream device. Once set, this flag may only be cleared when data for this point is received from the adjacent downstream device and the COMM_LOST flag received with that data is cleared.
REMOTE_FORCED	If set, the data value is overridden in a downstream reporting device. Only a non-originating device may set this flag. The flag is set when an overridden value is received. The REMOTE_FORCED flag shall be set in an object if either the REMOTE_FORCED or LOCAL_FORCED flags (see below) are set in an object received from a downstream device. An originating device may never set this bit. See flag description 11.1.6.1.1, NOTE 3.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Name	Functional description
LOCAL_FORCED	If set, the data value is overridden by the device that reports this flag as set. This may be due to the device operating in a diagnostic or temporary mode or due to human intervention. If the value is forced in a non-originating device **and** overridden in a downstream device, then the non-originating device shall set both REMOTE_FORCED and LOCAL_FORCED flags. See flag description 11.1.6.1.1, NOTE 3.
CHATTER_FILTER	Only applicable to single-bit binary input and double-bit input object groups. If set, the binary data value is presently changing between states at a sufficiently high enough rate to activate a chatter filter. The binary data value reported does not necessarily represent the actual state because the chatter filter may clamp the reported value to a single state during the time it is active. The purpose of the chatter filter is to suppress event reporting for binary and double-bit inputs that are experiencing a rapid series of state changes. The determination of what constitutes “chattering” is device-dependent. While a binary input is chattering, the originating device shall set the CHATTER_FILTER flag. When the chattering input again becomes stable, the originating device shall clear the flag and report the current state of the input. While CHATTER_FILTER is set, no events shall be generated for the binary input. Events are generated when the CHATTER_FILTER flag is set and cleared.
ROLLOVER	Only applicable to counter object groups. This flag is obsolete and should not be set in new designs. Information is presented here for historical reasons. There is no mechanism within DNP3 for the outstation to report the value at which counter rollover occurs (i.e., the maximum possible counter value). Hence outstations shall not set the ROLLOVER flag and master devices shall ignore the ROLLOVER flag. If polled data reporting is used, the master is responsible for polling counter data frequently enough to detect rollover.
OVER_RANGE	Only applicable to analog input, analog output status, and analog output event object groups. If set, the data object’s true value exceeds the valid measurement range of the object. See flag description 11.1.6.1.1, NOTE 4 for more details.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Name	Functional description
DISCONTINUITY	Only applicable to counter object groups. If set, the reported counter value cannot be compared against a prior value to obtain the correct count difference. The resetting of a counter through the use of freeze and clear commands, [FREEZE_CLEAR] and [FREEZE_CLEAR_NR], is a normal operation that shall not cause setting of the DISCONTINUITY flag. The flag is cleared after a new value has been acquired and transmitted to an upstream device.
REFERENCE_ERR	Only applicable to analog input, analog output status, and analog output event object groups. If set, the measurement process determined that the object's data value might not have the expected level of accuracy. For example: the reference signal used in the analog-to-digital conversion process is out of limits or a calculated value has been contaminated with noise.
STATE	Only applicable to single-bit binary input and binary output object groups. The flag's state indicates the state of the single-bit binary input or output point. STATE bits do not "flag" an exception condition per se. (See 11.1.6.1.1, NOTE 1.) They occupy a flag bit as a convenience in order to reduce the number of octets transmitted; however, the reported state values may be used to indicate error conditions.

11.1.6.1.1 Flag description notes

NOTE 1—The RESTART, COMM_LOST, LOCAL_FORCED, REMOTE_FORCED, CHATTER_FILTER, ROLLOVER, OVER_RANGE, DISCONTINUITY, and REFERENCE_ERR flags all indicate "exception conditions."

In normal operation, the ONLINE flag is set (1) and the exception condition flags are all cleared (0). Any other combination of the ONLINE and exception condition flags indicates that the data value might not correctly indicate the value of the corresponding field point.

NOTE 2—When an outstation reports a variation that has no flags, the master shall interpret this as if the flags were included with the ONLINE flag set and all the exception condition flags clear. If any other condition holds, then the outstation should report a variation with flags.

NOTE 3—For output status objects, the LOCAL_FORCED and REMOTE_FORCED flags indicate that the reported value has been overridden. This value might not correspond to the state of the output point. These flags **do not** indicate that a control was issued "locally" or "remotely" to the output point to set it to the reported state.

NOTE 4—Rules for setting the OVER_RANGE flag.

Some analog input data gathered by an outstation may inherently have a certain size, for example, devices that use 12-bit A/D converters. However, a master may request this data in a particular format such as a 16-bit or 32-bit integer. The following rules govern reporting this type of data:

— If a master requests a particular object variation, for example a 16-bit analog input, and the measured value of the data point within the outstation is within the range for the DNP3 variation (–32768 to 32767 for the 16-bit example), then the outstation reports the value without modification within the requested variation.

— For a data values stored within the device having lower resolution than the requested data size, this simply involves the sign extending the value to the requested size. Since DNP3 analog values are signed, sign extension is not considered a modification of the value. For example, a device internally saves analog measurements as 12-bit two's complement numbers; if the master requests a variation calling for a size of 16 or 32 bits, the outstation sign extends its 12-bit values for the response.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— If a master requests a 16-bit variation and the value of the data point is outside of the range –32768 (0x8000) to 32767 (0x7FFF), then the outstation reports the value as either –32768 (for a negative over-range) or 32767 (for a positive over-range) and it sets the OVER_RANGE flag of the object.

— If a master requests a 32-bit variation and the value of the data point is outside of the range –2147483648 (0x80000000) to 2147483647 (0x7FFFFFFF), then the outstation reports the value as either –2147483648 (for a negative over-range) or 2147483647 (for a positive over-range) and sets the OVER_RANGE flag of the object.

— If an input exceeds the range measurable by the hardware on the outstation, the outstation sets the OVER_RANGE flag of the object. It does not alter the value reported by its hardware. For example, an outstation with a 12-bit A/D converter shall set OVER_RANGE when the input exceeds the full scale limit of the converter, and the outstation shall report the converter's output as its maximum 2047 value in the analog input object.

— Floating-point data types shall report a value that is less than or equal to the measurement range minimum, or greater than or equal to the measurement range maximum, when the OVER_RANGE flag is set.

These rules are illustrated in Table 57. The outstation sets the OVER_RANGE flag in each case except in the example of 33 000 (decimal) requested as a 32-Bit Analog Input. This is a case of the second bullet applying but not the third bullet.

The term "full scale," as used in Table 57, refers to the input levels to an A/D converter that results in either the minimum or the maximum converted output value.

Table 57—Setting of OVER_RANGE flag examples

A/D converter type (min / max output value)	Input level	Value reported in 16-bit variations (hexadecimal)	Value reported in 32-bit variations (hexadecimal)
8 bits signed (-128 to 127)	input > full scale	0x007F	0x0000007F
	input < full scale	0xFF80	0xFFFFFF80
8 bits unsigned (0 to 255)	input > full scale	0x00FF	0x000000FF
	input < full scale	0x0000	0x00000000
12 bits signed (–2048 to 2047)	input > full scale	0x07FF	0x000007FF
	input < full scale	0xF800	0xFFFFF800
16 bits signed (–32768 to 32767)	input > full scale	0x7FFF	0x00007FFF
	input < full scale	0x8000	0xFFFF8000
32 bits signed (–2147483648 to 2147483647)	input > full scale	0x7FFF	0x7FFFFFFF
	input converts to value of 33 000 decimal	0x7FFF	0x00080E8[a]
	input < full scale	0x8000	0x80000000
12-bit signed (scaled to range of –16768 to 16752)[b]	input > full scale	0x7FF0	0x00007FF0
	input < full scale	0xFFFF	0xFFFF8000

[a]OVER_RANGE is not set in this case, but it is set in all other cases.
[b]In some devices, native analog ranges are "normalized" to fit into a 16-bit range and the measured value is scaled prior to being reported. For example, the output value of a 12-bit A/D converter could be scaled by multiplying by 16. This results in values in the range of –32 768 to 32 752 with steps of 16 between adjacent contiguous values. In cases where scaling is employed, the minimum and maximum scaled values are related to negative and positive "full-scale" values at the input. If the input quantity exceeds the full-scale input value, the respective minimum or maximum scaled value is reported and the OVER_RANGE flag is set.

NOTE Non-originating devices that obtain data from downstream devices by a method other than DNP3 communication should do their best to set the flag bits to the appropriate states.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

11.1.6.2 Interaction or combinations of flags

— Each flag indicates a separate condition. The setting or clearing of a flag does not necessarily cause other flags to set or clear. In particular: Clearing of the ONLINE flag shall not require that any other flag be set.

— The COMM_LOST flag indicates that communication has failed somewhere in the downstream path. This flag can only be set by a non-originating device. The reporting device shall not alter the object value or any other flag received from a downstream device if the communication is lost, except that the ONLINE flag may optionally be cleared. Thus, the value and other flags indicate the state of the input or output point when data was last collected prior to communication failure. A data object with the COMM_LOST flag set can be used as an indication of the last known value of the input or output if the other exception condition flags are not set.

— The REMOTE_FORCED and LOCAL_FORCED flags indicate that data has been overridden. The state of these flags shall not affect or alter the state of any other flag. In particular, the setting of these flags shall not clear the ONLINE flag or set the RESTART flag unless that is what the reporting device actually does to the data object.

— Setting the CHATTER_FILTER, ROLLOVER, OVER_RANGE, REFERENCE_ERR, and DISCONTINUITY flags shall not clear the ONLINE flag.

— If a device is capable of reporting data (communicating with an upstream device) prior to the clearing of the RESTART flag in the reported objects, then the ONLINE flag shall be cleared at initialization and the following conditions shall hold:

1) If the ONLINE flag is reported set while the RESTART flag is set, the object is operating and reporting its default start-up value, but it has not yet had its data updated from the field.

2) If the ONLINE flag is clear, the object has not completed its start-up process,and is still off-line.

11.1.6.3 Implementation rules

If the value and flags of an object are preserved through a cold start or restart, the RESTART flag of that object may optionally remain unchanged.

When the data object is first updated with field data, the RESTART flag shall be cleared.

If a device overrides the initial value of objects, then during the period between initialization and the time that the data object is first updated with field data, the objects may optionally set their LOCAL_FORCED flag to indicate that the value has been overridden.

11.1.6.4 Considerations for data concentrators (non-originating devices)

Data concentrators are non-originating devices acting as “reporting devices” that collect data from “originating devices” or other reporting devices. A data concentrator accepts data and flag values sent to it from a reporting device and passes these on to a master device. The data concentrator may alter some of the flags as follows:

— Unless otherwise specified, all flags are passed through unchanged.

— When the data concentrator undergoes a cold start, it shall set the RESTART flag and clear all other flags for all database points. These flags shall stay in this state until the first data update from the lower level reporting device; at which time, the value and flags shall be set to the state reported from

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

that downstream device. The data objects may optionally be initialized with the ONLINE and LOCAL_FORCED flags set as outlined in the preceding discussion. Alternatively, the data concentrator may choose to not respond until the first data update from the reporting device.

— If a downstream reporting device fails to communicate with the data concentrator, the data concentrator may set the COMM_LOST flag for all data objects from that reporting device. It may optionally also clear the ONLINE flag but shall not alter any other flag. When communication with the lower level device is established or restored, the data concentrator updates all flags to the state reported by the downstream device when data is received (subject to the following rule for handling the forced data flags).

— If the LOCAL_FORCED or REMOTE_FORCED flags are set for an object from a downstream reporting device, the data concentrator shall set the REMOTE_FORCED flag that it passes to the master. If the data concentrator overrides a data point's value, it shall set the LOCAL_FORCED flag; otherwise this flag shall be cleared.

11.1.6.5 Object groups having variations with flags and without flags

Some object groups have variations that include flags and other variations that do not. For these object groups, a device is permitted to determine if it shall report a variation with flags or a variation without flags. The variation without flags shall only be reported if the flag status for all data reported with that variation is ONLINE and none of the exception flag bits are set. If any data point is not ONLINE or has some exception condition flag being set, then a variation with flags shall be used to report that data. Hence, if a variation without flags is used, the receiving device shall interpret this to mean that the data is ONLINE with no exception conditions.

Note that the sending device may determine if flags should or should not be sent for object groups having variations with and without flags. Even if the master device requests a specific variation of an object, the outstation may respond with a different variation (with or without flags) as appropriate according to the condition stated earlier. Other attributes of a specific variation requested by a master, for example, 16-bit or 32-bit data, shall be observed by the outstation when responding.

11.1.7 Status codes

Many of the DNP3 objects return status codes. The codes common to multiple variations are listed here instead of repeating the same information on each of the object insert sheets.

11.1.7.1 Status codes for control-related objects

Status codes appear in the responses for control output commands associated with control blocks (object group 12), analog output blocks (object group 41), and output command events (object groups 13 and 43). Table 58 provides a list of valid status codes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 58 — Control-related status codes

Code number	Identifier name	Description
0	SUCCESS	Request accepted, initiated, or queued.
1	TIMEOUT	Request not accepted because the *operate* message was received after the *arm* timer timed out. The *arm* timer was started when the *select* operation for the same point was received.
2	NO_SELECT	Request not accepted because no previous matching *select* request exists. (An *operate* message was sent to activate an output that was not previously armed with a matching *select* message.)
3	FORMAT_ERROR	Request not accepted because there were formatting errors in the *control* request (either *select, operate,* or *direct operate*).
4	NOT_SUPPORTED	Request not accepted because a control operation is not supported for this point.
5	ALREADY_ACTIVE	Request not accepted, because the control queue is full or the point is already active.
6	HARDWARE_ERROR	Request not accepted because of control hardware problems.
7	LOCAL	Request not accepted because Local/Remote switch is in Local position.
8	TOO_MANY_OBJS	Request not accepted because too many objects appeared in the same request.
9	NOT_AUTHORIZED	Request not accepted because of insufficient authorization.
10	AUTOMATION_INHIBIT	Request not accepted because it was prevented or inhibited by a local automation process.
11	PROCESSING_LIMITED	Request not accepted because the device cannot process any more activities than are presently in progress.
12	OUT_OF_RANGE	Request not accepted because the value is outside the acceptable range permitted for this point.
13 to 125	RESERVED	Reserved for future use.
126	NON_PARTICIPATING	Sent in request messages indicating that the outstation shall not issue or perform the control operation.[35]
127	UNDEFINED	Request not accepted because of some other undefined reason.

11.1.7.2 Status codes for file-related objects

A status code appears in the response to each of the request types. Table 59 provides a list of valid status codes for all file transfer operations.

[35] Control status code 126, NON_PARTICIPATING, may be used as a test or "no-op." Specific control-related objects may have a further explanation for using this code. An outstation shall not reject requests with this status code or report parameter error in IIN2.2 unless there is some other reason to do so.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 59—File-related status codes

Code number	Identifier name	Description	Used with function codes
0	SUCCESS	The requested operation was successful.	2, 25, 26, 27, 30
1	PERMISSION_DENIED	Permission was denied due to improper authentication key, user name or password.	25, 27
2	INVALID_MODE	An unsupported or unknown operation mode was requested.	25
3	FILE_NOT_FOUND	The requested file does not exist.	25, 27, 28
4	FILE_LOCKED	The requested file is already in use by another user.	25, 27
5	TOO_MANY_OPEN	File could not be opened because the number of simultaneously opened files would be exceeded.	25
6	INVALID_HANDLE	There is no file opened with the handle in the request.	1, 2, 26, 30
7	WRITE_BLOCK_SIZE	The outstation is unable to negotiate a suitable write block size.	25
8	COMM_LOST	Communications were lost or cannot be established with the end device where the file resides.	1, 2, 25, 26, 27, 28, 30
9	CANNOT_ABORT	An abort request was unsuccessful because the outstation is unable or not programmed to abort, or the outstation knows that aborting the file would make it unusable.	30
10 to 15	RESERVED	Reserved for future use.	
16	NOT_OPENED	File handle does not reference an opened file.	26
17	HANDLE_EXPIRED	File closed due to inactivity timeout. This code is sent in a file transport status event object (object group 70, variation 6) when the timeout occurs.	None
18	BUFFER_OVERRUN	Too much file data was received for outstation to process.	2
19	FATAL	An error happened in the file processing that prevents any further activity with this file.	1, 2, 25, 26, 27, 28, 30
20	BLOCK_SEQ	The block number did not have the expected sequence number.	1, 2
21 to 254	RESERVED	Reserved for future use.	
255	UNDEFINED	Some other error not listed here occurred. Optional text explaining the error may appear in the octets following the status code.	1, 2, 25, 26, 27, 28, 30

11.1.8 Group number categories

DNP3 group numbers generally fall into categories as follows:

Device Attributes	group number 0
Binary Inputs	group numbers 1 to 9
Binary Outputs	group numbers 10 to 19
Counters	group numbers 20 to 29
Analog Inputs	group numbers 30 to 39

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Analog Outputs	group numbers 40 to 49
Time	group numbers 50 to 59
Class	group numbers 60 to 69
Files	group numbers 70 to 79
Devices	group numbers 80 to 82
Data Sets	group numbers 83 to 89
Applications	group numbers 90 to 99
Alternate Numerics	group numbers 100 to 109
Other	group numbers 110 and higher

Unused group and variations are reserved. Only the DNP User's Group may assign new numbers.

11.1.9 Point types

Many of the objects in the library are organized according to the traits that differentiate points of one type from the other point types. In DNP3, a *point* is a uniquely identifiable physical or logical entity. The term "point" applies to inputs like analogs, binaries, and counters and to outputs like analogs and binaries. A single analog input is an example of a point. It is associated with a specific measured signal or computed analog quantity. In addition to a value, an analog input may have a name, a scale factor, a deadband value for event detection, and other attributes.

A *point type* is a means of categorizing points having related characteristics, similar functionality, and relationship to physical hardware or logical space. Using the example from the previous paragraph, an analog input belongs to the analog input point type.

In most cases, each point type has one set of zero-based index numbers to uniquely identify the specific point instances within the point type.

This subclause discusses the characteristics and attributes for the point types that appear in this library.

11.1.9.1 Analog input point type

11.1.9.1.1 General description

The analog input point type is used to monitor and report the signed numeric values of physical or logical analog quantities.

Analog quantities generally have a continuous range of possible values, subject to quantization, that represent a physical amount. The quantities may be signed, such as positive and negative DC currents, or

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

they can represent a magnitude only such as an AC RMS voltage; however, all values are transported in DNP3 messages as signed quantities.

There are almost no restrictions placed upon the source of values for analog input points. The values can come from measuring field inputs or from values collected from other devices or from the result of a computation. Analog inputs can represent position information such as a transformer tap number. Another use is representing states; e.g., a negative value could indicate below limit, 0 could mean normal, and a positive, non-zero value could indicate above limit.

11.1.9.1.2 Analog input model

Figure 67 diagrams the analog input model supported by DNP3.

Figure 67—Analog input model

An analog input point always has a present value, which is the most recently measured, obtained, or computed value.

A device with analog input points may optionally generate analog input events when something of interest occurs. Examples include the present value changes by an amount which is "enough to be interesting," the present value crosses a threshold, or a specific time or time interval is reached. The value reported in an analog input event object is the present value of the analog input at the instant when the event is detected. It is permissible for applications to omit reporting intermediate values if more than one analog event for the same point is generated before being reported to the master; i.e., only the most recent event is reported.

An analog input point may optionally support frozen analog input values and frozen analog input events. This is useful for reporting the analog values existing in points from multiple outstations at the same instant in time. Typically freezing is requested at regular intervals appropriate for the system. Upon a freeze, the present value of the analog input point is copied to a separate variable. The act of freezing an analog input point can be initiated from the master via a freeze command within a DNP3 message or from a local, internal or external source. By doing this, the master can read the frozen values, which do not change between freeze commands, at a somewhat leisurely pace instead of attempting to simultaneously read all of the analog input points in all of its outstations, an act that may be impossible.

There shall be an underlying analog input existing within the point in order to report frozen analog values.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Freeze operations take two forms—freeze and freeze-and-clear. With a freeze-and-clear type operation the underlying analog input value is preset to a suitable, possibly non-zero number after its value is copied to the frozen analog variable. Two applications where freeze-and-clear would be useful are when the underlying analog input holds either a peak value (minimum or maximum) or an integrated value that is reinitialized by the clear action from a freeze-and-clear command.

If an analog input point is capable of generating frozen analog input events, a new event shall be created each time the analog input point is frozen.

If events are supported, a change in one of the object flags (see 11.1.6.1) shall trigger creation of an event.

11.1.9.1.3 Analog deadbands

One of the possible approaches for determining an "interesting value" is the use of a deadband. An analog input event is generated based on the difference of its current value and the value that was most recently queued as an event, when compared to a deadband value. There are two methods commonly used for detection of analog input events based on a deadband.

— Fixed Deadband. If the absolute value of the difference between the present value of an analog input point and the value that was most recently queued as an event for that point exceeds the deadband value, then an event is generated for that point.

— Integrating Deadband. The difference between the present value of an analog input point and the value that was most recently queued as an event for that point is integrated over time. An event is generated when the absolute value of the integral exceeds the deadband value.

DNP3 does not specify which algorithm shall be used for deadbanding, and outstation vendors may choose to implement any deadbanding method, or none on any point.

Deadband values may be downloaded via the protocol. Devices that support analog input reporting deadbands are not required to maintain the deadband values through a reset and may revert to default values immediately following the reset. Vendors of devices that are able to preserve updated deadband values through a reset should note this in their device profile.

11.1.9.1.4 Applicable DNP3 object groups

Table 60 shows which object groups are associated with analog input point types. A specific point index number associated with any of these group numbers always references the same point.

Table 60 — Analog input point type object groups

Group number	Used for
30	Reporting the present value
31	Reporting the frozen value
32	Reporting analog input events or changes to the flag bits
33	Reporting frozen analog input events
34	Reading and writing analog deadband values

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

11.1.9.1.5 Non-frozen and frozen data in the same message

Masters shall differentiate between static analog input data and static frozen analog input data returned in polls for class 0 data. It is important that the master update its database with the proper object and does not overwrite a frozen value with an un-frozen value or vice-versa.

In response to a class 0 poll, an outstation device shall report either

— The analog input value.

— The frozen analog input value.

— Both values; however, if the device is capable of reporting both, the point shall be configurable to report just one, the analog input value or the frozen analog input value.

11.1.9.2 Analog output point type

11.1.9.2.1 General description

The analog output point type is used to provide an analog output signal for controlling physical or logical quantities.

Analog outputs generally have a continuous range of possible values, subject to quantization, that represent a physical amount. Analog output values are transported in DNP3 messages using signed values.

Analog outputs are frequently used to set the desired operating level or position of a process or hardware mechanism; this is why DNP3 analog outputs are sometimes referred to as set-point outputs in other protocols. Uses include setting internal numbers, parameter values and physical outputs (e.g., voltage signals and loop currents).

Physical analog outputs are usually derived from digital-to-analog conversion hardware, whereas virtual or pseudo analog outputs are most often software variables.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

11.1.9.2.2 Analog output model

Figure 68—Analog output point type model

An analog output value is set with a DNP3 request message having one of the Application Layer control-related function codes. In some devices, the value can also be set by other sources within the outstation, such as a local manual operator control or another software process.

An analog output status value is used to monitor an analog output. The value returned shall represent the analog output value; it should **not** come from feedback of the process being controlled by the analog output value[36] (e.g., not from the output of the plant or controlled device in Figure 68). The choice of where to obtain the analog output status value is device dependent. Because of this, analog output status values do not need to exactly match the analog output value. The accuracy of the value returned is device dependent.

A device with analog output points may optionally generate events when something of interest occurs. Examples include the output value changes by an amount that is "enough to be interesting," the present value crosses a threshold, or a specific time or time interval is reached. The value reported in an analog output event object is the present value of the analog output at the instant when the event is detected.

If events are supported, a change in one of the object flags (see 11.1.6.1) shall trigger creation of an event.

[36] An analog **input** point is often used to monitor the controlled process's output. Correlations between output points and the analog input points are outside the scope of DNP3.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

An analog output point may use deadbands to determine when there is a change of output value sufficient to be of interest. See 11.1.9.1.3 for more information on deadbands.

A command from any source to set an analog output may also generate an event. Command events are used to notify the master whenever

— An outstation receives a control from another master or a data concentrator.

— A control request is made from an internal application.

— A control is issued by a local operator panel.

11.1.9.2.3 Applicable DNP3 objects

Table 61 shows which object groups are associated with analog output point types. A specific point index number associated with any of these group numbers always references the same point.

Table 61—Analog output point type object groups

Group number	Used for
40	Reporting present value of analog outputs
41	Controlling analog output values
42	Reporting changes to the analog output or flag bits
43	Reporting output points being commanded from any source

11.1.9.3 BCD point type

11.1.9.3.1 General description

DNP3 does not characterize BCD values other than to specify the binary-coded decimal format within the DNP3 objects. BCD points may be used for inputs or outputs. Devices may choose to provide BCD points as a parallel method of reading values from other already defined point types such as analog inputs or counters.

Because of the device dependency, the device shall also manage or coordinate point indexes associated with BCD points.

Users and vendors of equipment that support BCD point types should be aware that interoperability with another vendor's equipment may not be practical or possible because of the device dependencies.

11.1.9.3.2 Applicable DNP3 objects

Table 62 shows which object groups are associated with BCD point types.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 62—BCD point type object groups

Group number	Used for
101	To convey device-dependent values in BCD form.

11.1.9.4 Binary output point type

11.1.9.4.1 General description

The binary output point type is used to provide a digital on-off drive signal or a pulsed drive signal for controlling a real or pseudo output device.

There are three output models that apply to binary outputs, Activation, Complementary Latch, and Complementary Two-Output. A description of each is provided as follows.

11.1.9.4.2 Activation model

Figure 69 diagrams the binary output activation model supported by DNP3.

Figure 69—Activation model

The activation model has a single virtual or physical output. The purpose of this type output is to initiate an action (i.e., "cause it to happen"). Examples include *Initiate Test*, *Acknowledge Alarm*, and *Trip Breaker*.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Two binary output points (separate indexes) based on the activation model can be used as a pair to perform complementary functions—on-off, trip-close, raise-lower, etc.

11.1.9.4.3 Complementary latch model

Figure 70 diagrams the binary output complementary latch model supported by DNP3.

Figure 70—Complementary latch model

A complementary latch model has a single virtual or physical output that remains latched in an active or non-active state depending on which command is received. Examples include *illumination on-off, enable-disable*, and *auto-manual*.

A complementary latch model always has a meaningful output state and, therefore, always generates Binary Output status events.

11.1.9.4.4 Complementary, two-output model

Figure 71 diagrams the binary output complementary, two-output model supported by DNP3.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 71 —Complementary, two-output model

A complementary, two-output model has two virtual or physical outputs, named close and trip at a single index. One or the other output is set active momentarily depending on which command is received. Examples include *Run-Stop motor, Trip-Close breaker, Raise-Lower transformer tap,* and *Open-Close valve*.

11.1.9.4.5 Common features of models

In some devices, the output action can also be initiated by other sources within the outstation, such as a local manual operator control or another software process.

All models of a binary output point type can report the output drive state if the characteristics of the point are such that the drive retains a meaningful state for a significant amount of time after a control command (from DNP3 or another source) is issued to the point. This is shown as the Binary Output Status Value block in the upper right corner of each figure. The output status is **not** obtained from the state of the driven device.[37] If the output drive is stateless or does not change to a meaningful state for a significant time following a control command, then the output status is reported as 0.

All models of a binary output point type may generate two types of events.

[37] The state of a driven device, such as a circuit breaker, valve, or logical device, is often monitored with a binary input or double-bit binary input point. Correlation between the output point and the input point that monitors the driven output is outside the scope of DNP3.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— A Binary Output event is generated if the characteristics of the point are such that the drive retains a meaningful state for a significant amount of time after a control command is issued to the point (from DNP3 or another source) and the drive changes state. However, if the output drive is stateless or does not change to a meaningful state for a significant time following a control command, a Binary Output event is not generated. A device that supports Binary Output events shall generate an event if any of the object flags change. (See 11.1.6.1.)

— A Binary Output Command event is generated whenever a command is issued to the point from either a DNP3 command or another source.

11.1.9.4.6 Applicable DNP3 Objects

Table 63 shows which object groups are associated with binary output point types. A specific point index number associated with any of these group numbers always references the same point.

Table 63—Binary output point type object groups

Group number	Used for
10	Reporting the present output status
11	Reporting changes to the output status or flag bits
12	Issuing control commands
13	Reporting control command was issued regardless of it source

NOTE

The recommended approach for performing on-off and close-trip controls is to use the ***select*** and ***operate*** or the ***direct operate*** function codes with group 12 objects.

Note that using Application Layer function code ***write*** with a group 10 object to control a binary output point is not recommended because

— The response to a write command does not indicate whether the operation was successful.

— Devices may not support write operations to object group 10.

11.1.9.5 Counter point type

11.1.9.5.1 General description

The counter point type is used to monitor and report the values of monotonically increasing unsigned integer values.

Counting is a basic function appearing in many DNP3 devices. Typical examples include monitoring energy (KWHrs), fluid usage (liters) and the number of circuit breaker re-closures (operations count).

The values stored in counters may be obtained directly from field inputs such as electro-mechanical switches or from computed values.

Counter points often support *freezing* the counts. This is useful for reporting the count values existing in the points from multiple outstations at the same instant in time. Typically freezing is requested at regular

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

intervals appropriate for the system. Freezing involves copying the contents of a continuously running counter into a separate register or variable upon receipt of a freeze command from a DNP3 master or another source. By doing this, the master can read the frozen counts, which do not change between freeze commands, at a somewhat leisurely pace instead of attempting to simultaneously read all of the counter points in all of its outstations, an act that may be impossible.

11.1.9.5.2 Counter model

Figure 72 diagrams the binary counter model supported by DNP3.

Figure 72—Counter point type model

It is recommended that DNP3 counter points be based on modulo 65 536 or modulo 4 294 967 296 arithmetic and require either 16- or 32-bit unsigned binary values. During the counting process, the count always increases monotonically until the counter's maximum value, 65 535 or 4 294 967 295, respectively, is reached, at which time the count rolls over to zero and the counter continues counting. Rollover values other than 65 536 and 4 294 967 296 are permitted. Counters may be preset to a value, including 0, by DNP3 commands, such as freeze-and-clear or write requests, or from other sources within the device where they reside.

Freeze operations take two forms—freeze and freeze-and-clear. With a freeze-and-clear type operation, the count value is zeroed after its counts are copied to the frozen counter. Freeze operations may also come from other software processes or hardware within the outstation.

A device may configure counter support in any manner consistent with the following rules:

a) A binary counter **shall** exist for every counter point.

b) A frozen counter **may** exist for each binary counter (but not necessarily).

c) If a frozen counter exists, its index **shall** match that of the corresponding binary counter.

When a device restarts, frozen counter values may be undefined prior to processing the first freeze operation. Devices that maintain frozen counter values in non-volatile memory and that continue

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

accumulation during a loss-of-power condition may possibly maintain valid frozen counter values across a device restart operation.

Counters point types may optionally support events. The determination as to what makes a count significant to report as an event is vendor specific. As only one example, an event may be created if the difference between the present count and the count value in the last queued event exceeds a specific amount.

If the point supports frozen counter events, the point creates an event at every freeze. The value in a frozen counter event is the same value that is copied to the frozen count at the time of the freeze.

A device that supports counter events shall generate an event whenever any of the object flags change. (See 11.1.6.1.)

11.1.9.5.3 Implementation precedence

Implementation of a frozen count value, counter events, and frozen count events are optional; however, there shall be an underlying counter existing within the point in order to report that data. (See Figure 72.)

Frozen count events may only be implemented if the point implements a frozen counter. (See Figure 72.)

11.1.9.5.4 Positive and negative accumulations

There are counting applications where the totals can be positive or negative. An example is a co-generation facility that sometimes purchases energy and sometimes supplies it. When power flows from the facility, the accumulation might have a positive total, and when power is consumed by the plant, the accumulation is negative.

In cases where the accumulation has a mathematical sign, + or –, the use of two DNP3 counter points is required. One of the points is incremented when the polarity is positive and the other point is incremented when the polarity is negative. This maintains the rule that counters shall always increase monotonically and yet still provides the ability to report positive and negative accumulations.

11.1.9.5.5 Counts and frozen counts in the same message

Masters shall differentiate between static counter data and static frozen count data returned in polls for class 0 data. It is important that the master update its database with the proper object and does not overwrite a frozen value with an un-frozen value or vice versa.

In response to a class 0 poll, an outstation device shall report either

— The count value.

— The frozen count value.

— Both values; however, if the device is capable of reporting both, the point shall be configurable to report just one, the count value or the frozen count.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

11.1.9.5.6 Applicable DNP3 objects

Table 64 shows which object groups are associated with counter point types. A specific point index number associated with any of these group numbers always references the same point.

Table 64—Counter point type object groups

Group number	Used for
20	Reporting the count value
21	Reporting the frozen count value or changed flag bits
22	Reporting counter events
23	Reporting frozen counter events

11.1.9.6 Double-bit binary input point type

11.1.9.6.1 General description

Double-bit binary input points have two stable states, such as on and off, and an in-transit state. A fourth state can represent a condition that does not occur during normal operation.

Motor-operated power line switches and motor-operated valves are examples of field devices whose status monitoring benefits by using a double-bit input point. These devices have three states: open, closed, and transitioning between opened and closed (and vice versa). Typically, it requires seconds to move from one end position to the other. During the transition, there is a possibility of something going wrong and causing the device to stall; reporting the intermediate or transitioning state is advantageous because if it exists for too long, a problem may be indicated.

The double-bit binary input point type is suitable for bi-stable devices, like switches, where the field wiring utilizes form-C contacts instead of form-A. Form-C wiring requires two inputs. When a form-C device is at rest, one input is asserted and the other is not, but while the contact moves between one input and the other, and during contact bouncing, there is an intermediate condition. If the form-C contact is of type

— Break-before-make, then neither input is asserted during the intermediate condition (third state) and the fourth state indicates that both inputs are shorted together.

— Make-before-break, then both inputs are asserted during the intermediate condition (third state) and the fourth state indicates open circuited inputs (disconnected wiring).

Users are not prevented from using double-bit input points for other devices that have up to four states; however, they should not be used to represent two uncorrelated single-bit binary inputs.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

11.1.9.6.2 Double-bit binary input model

Figure 73 diagrams the double-bit binary input model supported by DNP3.

Figure 73—Double-bit binary input model

A double-bit input point is modeled as two single-bit binary inputs that are indivisibly linked together.

The four input states are defined in Table 65.

Table 65—Double-bit binary input states

State value (UINT2)	State name	Description	Corresponding state if a single-bit binary input were used instead
0	INTERMEDIATE	Transitioning between end conditions	—
1	DETERMINED_OFF	End condition, determined to be OFF	0
2	DETERMINED_ON	End condition, determined to be ON	1
3	INDETERMINATE	Abnormal or custom condition	—

DNP3 does not define whether the end conditions, DETERMINED_OFF and DETERMINED_ON, represent Opened-Closed, Raised-Lowered, On-Off, etc., as this is an implementation detail. It is recommended that outstation vendors provide a clear definition for users, and that masters provide configuration to accommodate either situation. Vendors shall explicitly state what each state represents if a double-bit binary input is used to monitor some other type of three or four-state device.

State 1 corresponds to what would be reported as a 0, and state 2 corresponds to what would be reported as a 1, if double-bit inputs were not available and a single-bit binary input were used instead.

The logical states listed previously are independent of the values read from the physical inputs. The outstation is responsible for mapping the physical inputs to the correct logical states.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A double-bit input point always has a current value, which is the most recently measured, obtained, or computed state.

A double-bit input point may optionally generate an event whenever the state changes. Only a single event is generated, not two events, if both inputs change simultaneously or within a device-dependent interval.

A device that supports double-bit binary input events shall generate an event whenever any of the object flags change. (See 11.1.6.1.)

11.1.9.6.3 Point space

Double-bit binary inputs occupy a different point space than do single-bit binary inputs. A double-bit binary input point and a single-bit binary input point having the same index do not necessarily represent the same physical or logical entity. DNP3 provides no mechanism to correlate double-bit points and single-bit binary input points even though it may be appropriate in certain situations. This does not prevent points from being correlated privately.

11.1.9.6.4 Applicable DNP3 objects

Table 66 shows which object groups are associated with double-bit input point types. A specific point index number associated with any of these group numbers always references the same point.

Table 66—Double-bit input point type object groups

Group number	Used for
3	Reporting present state value
4	Reporting double-bit binary input events and flag bit changes

11.1.9.7 Octet string point type

11.1.9.7.1 General description

Octet strings may be used to represent any block of 8-bit quantities. Often, they are used for ASCII strings, but they may also be used for non-printing strings, arbitrary-length coded data, or memory dumps. The content of an octet string is a local matter unless otherwise specified in the DNP3 documentation.

Reading and writing of 8-bit memory locations can be implemented using this object together with absolute addressing qualifiers. In this case there is no point index, and users shall need to creatively determine how to differentiate one memory region from another.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

11.1.9.7.2 Octet string model

Figure 74 diagrams the octet string model supported by DNP3.

Figure 74—Octet string model

The relationship between an octet string point index number and a physical or logical entity is a local matter.

Devices may create octet string events. The criteria for generating an octet string event is a local matter.

11.1.9.7.3 Applicable DNP3 objects

Table 67 shows which object groups are associated with octet string point types. A specific point index number associated with any of these group numbers always references the same point.

Table 67—Octet string point type object groups

Group number	Used for
110	To convey the present value
111	Reporting an octet string event

11.1.9.8 Single-bit binary input point type

11.1.9.8.1 General description

The single-bit binary input point type is used to monitor and report the state of internal or physical quantities that have only two possible values: 0 and 1.

There are almost no restrictions placed upon the source of values for single-bit binary input points. The values can come from sampling field inputs or from values collected from other devices or from the result of a Boolean computation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

11.1.9.8.2 Single-bit binary input model

Figure 75 diagrams the single-bit binary input model supported by DNP3.

Figure 75—Single-bit binary input point type model

A single-bit binary input point always has a current value, which is the most recently measured, obtained, or computed state.

DNP3 does not specify what a 0 or 1 state indicates, and vendors are permitted to assign either state to represent opened or closed, on or off, active or inactive, etc. Parameters specifying debounce timing, time-stamp accuracy and resolution, minimum on and off times, maximum toggle rates, and maximum number of changes per unit of time are outside the scope of DNP3—these are factors that vendors and users shall consider for the intended application.

A device that supports single-bit binary input events shall generate events whenever the single-bit binary input value changes state or whenever any of the object flags change. (See 11.1.6.1.)

11.1.9.8.3 Applicable DNP3 objects

Table 68 shows which object groups are associated with single-bit binary input point types. A specific point index number associated with any of these group numbers always references the same point.

Table 68—Single-bit binary input point type object groups

Group number	Used for
1	Reporting the present value of a single-bit binary input
2	Reporting single-bit binary input events and flag bit changes

11.1.9.9 Virtual terminal point type

11.1.9.9.1 General description

Many IEDs have separate, non-DNP3, serial interfaces used for configuration, diagnostics, and other ancillary tasks. These utility ports are often set up for connection to a dumb terminal or PC, and the protocol on these ports is often an ASCII command set customized for the particular IED. VT points

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

provide a somewhat transparent means to communicate with these ports by sharing the existing DNP3 bandwidth. This scheme eliminates the need for a parallel set of communication links (Figure 76).

Figure 76—Virtual terminal conceptual model

Information between the terminal equipment or process at each end are transported over what is called a virtual communication channel. Each virtual communication channel is assigned a virtual port number.

The procedure for communicating with virtual terminal points is as follows. Master devices transmit data to outstation devices by writing one or more group 112 objects using the DNP3 point index number to specify the virtual port number.

Outstations return information to the master by responding to ***read*** requests for object group 113, responding to ***read*** requests for the configured event class, or transmitting an unsolicited response message.

The Device Profile Document shall list the point index number(s) assigned to virtual terminals.

11.1.9.9.2 Applicable DNP3 objects

Table 69 shows which object groups are associated with virtual terminal point types. A specific point index number associated with any of these group numbers always references the same point.

Table 69—Virtual terminal point type object groups

Group number	Used for
112	Conveying data to the command interpreter at the outstation
113	Conveying data from the command interpreter at the outstation

11.1.10 Insert formats

Inserts are pages that form a part of the specification and describe the object group and variation numbers.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Each insert has a header at the top of the page that specifies the object's group and variation names and numbers. It also contains a type field that specifies the object's type category (see 11.1.5).

DNP3 Object Library		Group:	30
		Variation:	1
Group Name:	Analog Input	Type:	Static
Variation Name:	32-bit with flag		

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

12. DNP3 object library—parsing codes

12.1 Data object library—part three

12.1.1 Subset parsing codes

The tables that follow summarize the combinations of groups, variations, function codes, and qualifier codes that apply to DNP3 subset levels. A master or outstation shall be able to parse the codes shown in the tables for the DNP3 subset level it claims. Having the ability to parse does not require a device to have points, attributes, etc. that would be transferred by objects of the particular group and variation. A device may optionally parse groups, variations, function codes, and qualifiers that are in addition to the minimum requirements of the subset level it claims. Subclause 12.1.2 provides suggestions for using DNP3 objects not described in the subset parsing tables. Every device shall be able to limit the objects, function codes, and qualifier codes that it transmits to those required by the subset level of the receiving master or outstation device.

12.1.1.1 How to interpret the subset parsing tables

12.1.1.1.1 Request and response column independence

The reader should consider each of the following subset parsing tables as a merger of two independent tables: a request table and a response table that are combined for convenience. When interpreting the tables, use the columns described in 12.1.1.1.2 or 12.1.1.1.3.

The reader shall not assume a relationship between the table entries in both the request and response columns (for the same row). In some cases, DNP3 responses indicated in the Response column do apply to corresponding DNP3 requests indicated in the Request column, but this is not always true. In most circumstances, DNP3 responses are returned as a result of requests not listed in the same row.

12.1.1.1.2 Interpreting the subset parsing tables for outstation devices

Find the group and variation number and then the subset level of the outstation. The outstation device shall be capable of parsing all of the function and qualifier codes listed under the Request column. For greater flexibility, vendors may choose to implement a device so that it parses function and qualifier codes in higher subset levels, or from Table 70 through Table 100. The outstation device may only transmit function and qualifier codes listed under the Response column to masters based on the subset level of the master. The device may transmit the function and qualifier codes in other subset levels or from 12.1.2.2, if the master supports them.

Where the Request column is indicated with "—" (dash) marks, support for the group and variation (indicated on that row) is not required for the given subset levels. Where the Request column is shaded, the table row is not applicable to outstation parsing. An example of this is a table row representing an unsolicited response for master parsing.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

12.1.1.1.3 Interpreting the subset parsing tables for master devices

Find the group and variation number and then the subset level of the master. The master device shall be capable of parsing all the function and qualifier codes listed under the Response column. The master device may only transmit to outstations function and qualifier codes listed under the Request column, based on the subset level of the outstation. The master device may transmit function and qualifier codes in higher subset levels or from Table 70 through Table 100, if the outstation supports them.

Where the Response column is indicated with "—" (dash) marks, support for the group and variation (indicated on that row) is not required for the given subset levels. Where the Response column is shaded, the table row is not applicable to master parsing. An example of this is a table row representing variation 0 of an object for outstation parsing.

12.1.1.2 Subset parsing tables

Support for all objects of group 0 became mandatory for subset level 4 conformant devices from DNP3-2007.

Support for all objects of group 0 became mandatory for subset level 3 conformant devices from DNP3-2009. In addition it became mandatory for subset levels 3 and 4 to support the write function code on those variations identified in Table 70.

Support for some specific variations of group 0 objects became mandatory for subset level 1 and 2 conformant devices from DNP3-2009.

The parsing information in Table 70 applies to DNP3 group 0 attribute objects for index 0. Parsing rules for user-specific group 0 attribute sets (indexes 1 and above) are device dependent.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 70—g0 device attribute objects

Group	Variation	Subset levels					Request (outstation must parse)		Response (master shall parse)	
		1	2	3	4		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
0	211 –239			×	×		1 (READ)	00	129 (RESPONSE)	00,17
0	240			×	×		1 (READ)	00	129 (RESPONSE)	00,17
0	240			×	×		2 (WRITE)[a]	00		
0	241			×	×		1 (READ)	00	129 (RESPONSE)	00,17
0	242 –243	×	×	×	×		1 (READ)	00	129 (RESPONSE)	00,17
0	245			×	×		1 (READ)	00	129 (RESPONSE)	00,17
0	246	×	×	×	×		1 (READ)	00	129 (RESPONSE)	00,17
0	247			×	×		1 (READ)	00	129 (RESPONSE)	00,17
0	245 –247			×	×		2 (WRITE)[a]	00		
0	248	×	×	×	×		1 (READ)	00	129 (RESPONSE)	00,17
0	249			×	×		1 (READ)	00	129 (RESPONSE)	00,17
0	250	×	×	×	×		1 (READ)	00	129 (RESPONSE)	00,17
0	251			×	×		1 (READ)	00	129 (RESPONSE)	00,17
0	252	×	×	×	×		1 (READ)	00	129 (RESPONSE)	00,17
0	254	×	×	×	×		1 (READ)	00,06		
0	255	×	×	×	×		1 (READ)	00,06	129 (RESPONSE)	00,5B (Cnt=1)

[a] WRITE function codes in this table are shown as examples of attributes that may be writeable in subset level 1 and 2 conformant devices. Actual usage at subset levels 1 and 2 is device dependent. A master may read group 0, variation 255 to determine which group 0 attributes are writeable for a particular device.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 71—g1 binary input static objects

Group	Variation	Subset levels					Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
1	0	×					—	—		
1	0		×				1 (READ)	06		
1	0			×	×		1 (READ)	00,01,06		
1	0			×	×		22 (ASSIGN_CLASS)	00,01,06		
1	1	×	×				—	—	129 (RESPONSE)	00,01
1	1			×	×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
1	2	×	×				—	—	129 (RESPONSE)	00,01
1	2			×	×		1 (READ)	00,01,06	129 (RESPONSE)	00,01

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 72—g2 binary input event objects

Group	Variation	Subset levels				Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4	Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
2	0	×				—	—		
2	0		×	×	×	1 (READ)	06,07,08		
2	1	×				—	—	129 (RESPONSE)	17,28
2	1		×	×	×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
2	1	×	×	×	×			130 (UNSOL_RESP)	17,28
2	2	×				—	—	129 (RESPONSE)	17,28
2	2		×	×	×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
2	2	×	×	×	×			130 (UNSOL_RESP)	17,28
2	3	×				—	—	129 (RESPONSE)	17,28
2	3		×	×	×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
2	3	×	×	×	×			130 (UNSOL_RESP)	17,28

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 73—g3 double-bit binary input static objects

Group	Variation	Subset levels					Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
3	0	×	×	×			—	—		
3	0				×		1 (READ)	00,01,06		
3	0				×		22 (ASSIGN_CLASS)	00,01,06		
3	1	×	×	×			—	—	—	—
3	1				×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
3	2	×	×	×			—	—	—	—
3	2				×		1 (READ)	00,01,06	129 (RESPONSE)	00,01

Table 74—g4 double-bit binary input event objects

Group	Variation	Subset levels					Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
4	0	×	×	×			—	—		
4	0				×		1 (READ)	06,07,08		
4	1	×	×	×			—	—	—	—
4	1				×		1 (READ)	06,07,08	129 (RESPONSE)	17,28
4	1				×				130 (UNSOL_RESP)	17,28
4	2	×	×	×			—	—	—	—
4	2				×		1 (READ)	06,07,08	129 (RESPONSE)	17,28
4	2				×				130 (UNSOL_RESP)	17,28
4	3	×	×	×			—	—	—	—
4	3				×		1 (READ)	06,07,08	129 (RESPONSE)	17,28
4	3				×				130 (UNSOL_RESP)	17,28

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 75—g10 binary output static objects

Group	Variation	Subset levels					Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
10	0	×	×				1 (READ)	06		
10	0			×			1 (READ)	00,01,06		
10	0				×		22 (ASSIGN_CLASS)	00,01,06		
10	1	×	×	×	×		—	—	—	—
10	2	×	×				—	—	129 (RESPONSE)	00,01
10	2			×	×		1 (READ)	00,01,06	129 (RESPONSE)	00,01

Table 76—g11 binary output event objects

Group	Variation	Subset levels					Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
11	0	×	×	×			—	—		
11	0				×		1 (READ)	06,07,08		
11	1	×	×	×			—	—	—	—
11	1				×		1 (READ)	06,07,08	129 (RESPONSE)	17,28
11	1				×				130 (UNSOL_RESP)	17,28
11	2	×	×	×			—	—	—	—
11	2				×		1 (READ)	06,07,08	129 (RESPONSE)	17,28
11	2				×				130 (UNSOL_RESP)	17,28

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 77—g12 binary output command objects

Group	Variation	Subset levels					Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
12	0	×	×	×			—	—		
12	0				×		22 (ASSIGN_CLASS)	00,01,06		
12	1	×	×	×	×		3 (SELECT)	17,28	129 (RESPONSE)	Echo request
12	1	×	×	×	×		4 (OPERATE)	17,28	129 (RESPONSE)	Echo request
12	1	×	×	×	×		5 (DIRECT_OPERATE)	17,28	129 (RESPONSE)	Echo request
12	1	×	×	×	×		6 (DIRECT_OPERATE_NR)	17,28		
12	2	×	×	×	×		—	—	—	—
12	3	×	×	×	×		—	—	—	—

Table 78—g13 binary output command event objects

Group	Variation	Subset levels					Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
13	0	×	×	×			—	—		
13	0				×		1 (READ)	06,07,08		
13	1	×	×	×			—	—	—	—
13	1				×		1 (READ)	06,07,08	129 (RESPONSE)	17,28
13	1				×				130 (UNSOL_RESP)	17,28
13	2	×	×	×			—	—	—	—
13	2				×		1 (READ)	06,07,08	129 (RESPONSE)	17,28
13	2				×				130 (UNSOL_RESP)	17,28

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 79—g20 counter static objects

Group	Variation	Subset levels					Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
20	0	×					—	—		
20	0		×				1 (READ)	06		
20	0		×				7 (IMMED_FREEZE)	06		
20	0		×				8 (IMMED_FREEZE_NR)	06		
20	0		×				9 (FREEZE_CLEAR)	06		
20	0		×				10 (FREEZE_CLEAR_NR)	06		
20	0			×	×		1 (READ)	00,01,06		
20	0			×	×		7 (IMMED_FREEZE)	00,01,06		
20	0			×	×		8 (IMMED_FREEZE_NR)	00,01,06		
20	0			×	×		9 (FREEZE_CLEAR)	00,01,06		
20	0			×	×		10 (FREEZE_CLEAR_NR)	00,01,06		
20	0			×	×		22 (ASSIGN_CLASS)	00,01,06		
20	1	×	×				—	—	129 (RESPONSE)	00,01
20	1			×	×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
20	2	×	×				—	—	129 (RESPONSE)	00,01
20	2			×	×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
20	5	×	×				—	—	129 (RESPONSE)	00,01
20	5			×	×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
20	6	×	×				—	—	129 (RESPONSE)	00,01
20	6			×	×		1 (READ)	00,01,06	129 (RESPONSE)	00,01

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 80—g21 frozen counter static objects

Group	Variation	Subset levels					Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
21	0	×					—	—		
21	0		×				1 (READ)	06		
21	0			×	×		1 (READ)	00,01,06		
21	0			×	×		22 (ASSIGN_CLASS)	00,01,06		
21	1	×					—	—	—	—
21	1		×				—	—	129 (RESPONSE)	00,01
21	1			×	×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
21	2	×					—	—	—	—
21	2		×				—	—	129 (RESPONSE)	00,01
21	2			×	×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
21	5	×	×	×			—	—	—	—
21	5				×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
21	6	×	×	×			—	—	—	—
21	6				×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
21	9	×					—	—	—	—
21	9		×				—	—	129 (RESPONSE)	00,01
21	9			×	×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
21	10	×					—	—	—	—
21	10		×				—	—	129 (RESPONSE)	00,01
21	10			×	×		1 (READ)	00,01,06	129 (RESPONSE)	00,01

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 81—g22 counter event objects

Group	Variation	Subset levels				Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4	Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
22	0	×				—	—		
22	0		×	×	×	1 (READ)	06,07,08		
22	1	×	×			—	—	129 (RESPONSE)	17,28
22	1			×	×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
22	1	×	×	×	×			130 (UNSOL_RESP)	17,28
22	2	×	×			—	—	129 (RESPONSE)	17,28
22	2			×	×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
22	2	×	×	×	×			130 (UNSOL_RESP)	17,28
22	5	×	×	×	×	—	—	—	—
22	6	×	×	×	×	—	—	—	—

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 82—g23 frozen change event objects

Group	Variation	Subset levels				Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4	Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
23	0	×	×			—	—		
23	0			×	×	1 (READ)	06,07,08		
23	1	×	×			—	—	—	—
23	1			×	×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
23	1			×	×			130 (UNSOL_RESP)	17,28
23	2	×	×			—	—	—	—
23	2			×	×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
23	2			×	×			130 (UNSOL_RESP)	17,28
23	5	×	×	×		—	—	—	—
23	5				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
23	5				×			130 (UNSOL_RESP)	17,28
23	6	×	×	×		—	—	—	—
23	6				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
23	6				×			130 (UNSOL_RESP)	17,28

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 83—g30 analog input static objects

Group	Variation	Subset levels					Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
30	0	×					—	—		
30	0		×				1 (READ)	06		
30	0			×	×		1 (READ)	00,01,06		
30	0			×	×		22 (ASSIGN_CLASS)	00,01,06		
30	1	×	×				—	—	129 (RESPONSE)	00,01
30	1			×	×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
30	2	×	×				—	—	129 (RESPONSE)	00,01
30	2			×	×		1	00,01,06	129 (RESPONSE)	00,01
30	3	×	×				—	—	129 (RESPONSE)	00,01
30	3			×	×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
30	4	×	×				—	—	129 (RESPONSE)	00,01
30	4			×	×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
30	5	×	×	×			—	—	—	—
30	5				×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
30	6	×	×	×	×		—	—	—	—

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 84—g31 frozen analog input static objects

Group	Variation	Subset levels					Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
31	0	×	×	×	×		—	—		
31	1	×	×	×	×		—	—	—	—
31	2	×	×	×	×		—	—	—	—
31	3	×	×	×	×		—	—	—	—
31	4	×	×	×	×		—	—	—	—
31	5	×	×	×	×		—	—	—	—
31	6	×	×	×	×		—	—	—	—
31	7	×	×	×	×		—	—	—	—
31	8	×	×	×	×		—	—	—	—

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 85—g32 analog input event objects

Group	Variation	Subset levels				Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4	Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
32	0	×				—	—		
32	0		×	×	×	1 (READ)	06,07,08		
32	1	×	×			—	—	129 (RESPONSE)	17,28
32	1			×	×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
32	1	×	×	×	×			130 (UNSOL_RESP)	17,28
32	2	×	×			—	—	129 (RESPONSE)	17,28
32	2			×	×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
32	2	×	×	×	×			130 (UNSOL_RESP)	17,28
32	3	×	×	×		—	—	—	—
32	3				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
32	3				×			130 (UNSOL_RESP)	17,28
32	4	×	×	×		—	—	—	—
32	4				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
32	4				×			130 (UNSOL_RESP)	17,28
32	5	×	×	×		—	—	—	—
32	5				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
32	5				×			130 (UNSOL_RESP)	17,28
32	6	×	×	×	×	—	—	—	—
32	7	×	×	×		—	—	—	—
32	7				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
32	7				×			130 (UNSOL_RESP)	17,28
32	8	×	×	×	×	—	—	—	—

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 86—g33 frozen analog input event objects

Group	Variation	Subset levels				Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4	Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
33	0	×	×	×	×	—	—		
33	1	×	×	×	×	—	—	—	—
33	2	×	×	×	×	—	—	—	—
33	3	×	×	×	×	—	—	—	—
33	4	×	×	×	×	—	—	—	—
33	5	×	×	×	×	—	—	—	—
33	6	×	×	×	×	—	—	—	—
33	7	×	×	×	×	—	—	—	—
33	8	×	×	×	×	—	—	—	—

Table 87—g34 analog input reporting deadband objects

Group	Variation	Subset levels				Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4	Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
34	0	×	×	×		—	—		
34	0				×	1 (READ)	00,01, 06		
34	1	×	×	×		—	—	—	—
34	1				×	1 (READ)	00,01, 06	129 (RESPONSE)	00,01
34	1				×	2 (WRITE)	00,01,06	—	—
34	2	×	×	×		—	—	—	—
34	2				×	1 (READ)	00,01, 06	129 (RESPONSE)	00,01
34	2				×	2 (WRITE)	00,01,06	—	—
34	3	×	×	×		—	—	—	—
34	3				×	1 (READ)	00,01, 06	129 (RESPONSE)	00,01
34	3				×	2 (WRITE)	00,01,06	—	—

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 88—g40 analog output status objects

Group	Variation	Subset levels					Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
40	0	×	×				1 (READ)	06		
40	0			×	×		1 (READ)	00,01,06		
40	0				×		22 (ASSIGN_CLASS)	00,01,06		
40	1	×	×				—	—	—	—
40	1			×	×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
40	2	×	×				—	—	129 (RESPONSE)	00,01
40	2			×	×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
40	3	×	×	×			—	—	—	—
40	3				×		1 (READ)	00,01,06	129 (RESPONSE)	00,01
40	4	×	×	×	×		—	—	—	—

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 89—g41 analog output command objects

Group	Variation	Subset levels					Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
41	0	×	×	×			—	—		
41	0				×		22 (ASSIGN_CLASS)	00,01,06		
41	1	×	×				—	—	—	—
41	1			×	×		3 (SELECT)	17,28	129 (RESPONSE)	Echo request
41	1			×	×		4 (OPERATE)	17,28	129 (RESPONSE)	Echo request
41	1			×	×		5 (DIRECT_OPERATE)	17,28	129 (RESPONSE)	Echo request
41	1			×	×		6 (DIRECT_OPERATE_NR)	17,28		
41	2	×	×	×	×		3 (SELECT)	17,28	129 (RESPONSE)	Echo request
41	2	×	×	×	×		4 (OPERATE)	17,28	129 (RESPONSE)	Echo request
41	2	×	×	×	×		5 (DIRECT_OPERATE)	17,28	129 (RESPONSE)	Echo request
41	2	×	×	×	×		6 (DIRECT_OPERATE_NR)	17,28		
41	3	×	×	×			—	—	—	—
41	3				×		3 (SELECT)	17,28	129 (RESPONSE)	Echo request
41	3				×		4 (OPERATE)	17,28	129 (RESPONSE)	Echo request
41	3				×		5 (DIRECT_OPERATE))	17,28	129 (RESPONSE)	Echo request
41	3				×		6 (DIRECT_OPERATE_NR)	17,28		
41	4	×	×	×	×		—	—	—	—

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 90—g42 analog output event objects

Group	Variation	Subset levels				Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4	Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
42	0	×	×	×		—	—		
42	0				×	1 (READ)	06,07,08		
42	1	×	×	×		—	—	—	—
42	1				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
42	1				×			130 (UNSOL_RESP)	17,28
42	2	×	×	×		—	—	—	—
42	2				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
42	2				×			130 (UNSOL_RESP)	17,28
42	3	×	×	×		—	—	—	—
42	3				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
42	3				×			130 (UNSOL_RESP)	17,28
42	4	×	×	×		—	—	—	—
42	4				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
42	4				×			130 (UNSOL_RESP)	17,28
42	5	×	×	×		—	—	—	—
42	5				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
42	5				×			130 (UNSOL_RESP)	17,28
42	6	×	×	×	×	—	—	—	—
42	7	×	×	×		—	—	—	—
42	7				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
42	7				×			130 (UNSOL_RESP)	17,28
42	8	×	×	×	×	—	—	—	—

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 91—g43 analog output command event objects

Group	Variation	Subset levels				Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4	Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
43	0	×	×	×		—	—		
43	0				×	1 (READ)	06,07,08		
43	1	×	×	×		—	—	—	—
43	1				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
43	1				×			130 (UNSOL_RESP)	17,28
43	2	×	×	×		—	—	—	—
43	2				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
43	2				×			130 (UNSOL_RESP)	17,28
43	3	×	×	×		—	—	—	—
43	3				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
43	3				×			130 (UNSOL_RESP)	17,28
43	4	×	×	×		—	—	—	—
43	4				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
43	4				×			130 (UNSOL_RESP)	17,28
43	5	×	×	×		—	—	—	—
43	5				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
43	5				×			130 (UNSOL_RESP)	17,28
43	6	×	×	×	×	—	—	—	—
43	7	×	×	×		—	—	—	—
43	7				×	1 (READ)	06,07,08	129 (RESPONSE)	17,28
43	7				×			130 (UNSOL_RESP)	17,28
43	8	×	×	×	×	—	—	—	—

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 92—g50–g52 time information objects

Group	Variation	Subset levels				Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4	Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
50	1			×	×	1 (READ)	07 (Qty = 1)	129 (RESPONSE)	07 (Qty = 1)
50	1	×	×	×	×	2 (WRITE)	07 (Qty = 1)		
50	2	×	×	×	×	—	—		
50	3	×	×	×	×	—	—		
51	1	×	×	×	×	—	—	129 (RESPONSE)	07 (Qty = 1)
51	1	×	×	×	×			130 (UNSOL_RESP)	07 (Qty = 1)
51	2	×	×	×	×	—	—	129 (RESPONSE)	07 (Qty = 1)
51	2	×	×	×	×			130 (UNSOL_RESP)	07 (Qty = 1)
52	1	×	×	×	×	—	—	129 (RESPONSE)	07 (Qty = 1)
52	2	×	×	×	×	—	—	129 (RESPONSE)	07 (Qty = 1)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 93—g60 class information objects

Group	Variation	Subset levels				Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4	Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
60	1	×	×	×	×	1 (READ)	06		
60	2	×	×	×	×	1 (READ)	06,07,08		
60	2			×	×	20 (ENABLE_UNSOLICITED)	06		
60	2			×	×	21 (DISABLE_UNSOLICITED)	06		
60	2			×	×	22 (ASSIGN_CLASS)	06		
60	3	×	×	×	×	1 (READ)	06,07,08		
60	3			×	×	20 (ENABLE_UNSOLICITED)	06		
60	3			×	×	21 (DISABLE_UNSOLICITED)	06		
60	3			×	×	22 (ASSIGN_CLASS)	06		
60	4	×	×	×	×	1 (READ)	06,07,08		
60	4			×	×	20 (ENABLE_UNSOLICITED)	06		
60	4			×	×	21 (DISABLE_UNSOLICITED)	06		
60	4			×	×	22 (ASSIGN_CLASS)	06		

Table 94—g70 file objects

Group	Variation	Subset levels				Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4	Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
70	2	×	×	×	×	—	—	—	—
70	3	×	×	×	×	—	—	—	—
70	4	×	×	×	×	—	—	—	—
70	5	×	×	×	×	—	—	—	—
70	6	×	×	×	×	—	—	—	—
70	7	×	×	×	×	—	—	—	—
70	8	×	×	×	×	—	—	—	—

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 95—g80–g83 information objects

Group	Variation	Subset levels					Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
80	1			×	×		1 (READ)	00,01	129 (RESPONSE)	00,01
80	1	×	×	×	×		2 (WRITE)	00 (Index = 7)		
81	1	×	×	×	×		—	—	—	—
82	1	×	×	×	×		—	—	—	—
83	1	×	×	×	×		—	—	—	—
83	2	×	×	×	×		—	—	—	—

Table 96—g85–g88 data set objects

Group	Variation	Subset levels					Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4	op	Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
85	0	×	×	×	×		—	—		
85	1	×	×	×	×		—	—	—	—
86	0	×	×	×	×		—	—		
86	1	×	×	×	×		—	—	—	—
86	2	×	×	×	×		—	—	—	—
86	3	×	×	×	×		—	—	—	—
87	0	×	×	×	×		—	—		
87	1	×	×	×	×		—	—	—	—
88	0	×	×	×	×		—	—		
88	1	×	×	×	×		—	—	—	—

Table 97—g90 application & status of operation information objects

Group	Variation	Subset levels					Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
90	1	×	×	×	×		—	—	—	—

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 98 —g101–g102 numeric static objects

Group	Variation	Subset levels				Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4	Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
101	1	×	×	×	×	—	—	—	—
101	2	×	×	×	×	—	—	—	—
101	3	×	×	×	×	—	—	—	—
102	1	×	×	×	×	—	—	—	—

Table 99 —g110–g113 string & virtual terminal static & event objects

Group	Variation (length)	Subset levels				Request (outstation shall parse)		Response (master shall parse)	
		1	2	3	4	Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
110	1–255	×	×	×	×	—	—	—	—
111	1–255	×	×	×	×	—	—	—	—
112	1–255	×	×	×	×	—	—	—	—
113	0	×	×	×	×	—	—		
113	1–255	×	×	×	×	—	—	—	—

Table 100 —Function codes not used with objects

Subset levels				Request (outstation shall parse)	Response (master shall parse)
1	2	3	4	Function codes (decimal)	Function codes (decimal)
×	×	×	×	0 (CONFIRM)	
×	×	×	×	13 (COLD_RESTART)	
×	×	×	×	23 (DELAY_MEASUREMENT)	

12.1.2 Parsing guidelines

Table 101 through Table 130 are intended as guidelines for usage of groups, variations, function codes, and qualifier codes that are outside of the minimum requirements of DNP3 subset definitions. For completeness, the tables that follow also include the information from the subset parsing tables in the previous subclause.

These tables do not provide an exhaustive list of all valid DNP3 object / function code / qualifier combinations. Devices may implement different combinations to those shown in the tables and still use DNP3 in a valid way.

Having the ability to parse does not require a device to have points, attributes, etc. that would be transferred by objects of the particular group and variation. Every device shall be able to limit the objects, function

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

codes, and qualifier codes that it transmits to those required by the subset level of the receiving master or outstation device.

12.1.2.1 How to interpret the parsing guideline tables

12.1.2.1.1 Request and response column independence

The reader should consider each of the following tables as a merger of two independent tables: a request table and a response table that are combined for convenience. When interpreting the tables, use the columns described in 12.1.2.1.2 or 12.1.2.1.3.

The reader shall not assume a relationship between the table entries in both the request and response columns (for the same row). In some cases, DNP3 responses indicated in the response function code and qualifier columns do apply to corresponding DNP3 requests indicated in the request function code and qualifier columns, but this is not always true. In most circumstances, DNP3 responses are returned as a result of requests not listed in the same row.

12.1.2.1.2 Interpreting the parsing guideline tables for outstation devices

Find the group and variation number. The outstation device may parse some or all of the function and qualifier codes listed under the Request column. The outstation device may transmit function and qualifier codes as listed under the Response column.

Where the Request column is shaded, the table row is not applicable to outstation parsing. An example of this is a table row representing an unsolicited response for master parsing.

12.1.2.1.3 Interpreting the parsing guideline tables for master devices

Find the group and variation number. The master device may parse some or all of the function and qualifier codes listed under the Response column. The master device may transmit to outstations function and qualifier codes listed under the Request column if the outstation supports them.

Where the Response column is shaded, the table row is not applicable to master parsing. An example of this is a table row representing variation 0 of an object for outstation parsing.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

12.1.2.2 Parsing guideline tables

Table 101 —g0 device attribute objects

Group	Variation	Request (outstation shall parse)		Response (master shall parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
0	211 –239	1 (READ)	00	129 (RESPONSE)	00,17
0	240	1 (READ)	00	129 (RESPONSE)	00,17
0	240	2 (WRITE) [1]	00		
0	241 –243	1 (READ)	00	129 (RESPONSE)	00,17
0	245 –247	1 (READ)	00	129 (RESPONSE)	00,17
0	245 –247	2 (WRITE) [1]	00		
0	248 –252	1 (READ)	00	129 (RESPONSE)	00,17
0	254	1 (READ)	00,06		
0	255	1 (READ)	00,06	129 (RESPONSE)	00,5B (Cnt=1)

Table 102 —g1 binary input static objects

Group	Variation	Request (outstation shall parse)		Response (master shall parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
1	0	1 (READ)	00,01,06,17,28		
1	0	22 (ASSIGN_CLASS)	00,01,06,17,28		
1	1	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
1	2	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 103 —g2 binary input event objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
2	0	1 (READ)	06,07,08		
2	1	1 (READ)	06,07,08	129 (RESPONSE)	17,28
2	1			130 (UNSOL_RESP)	17,28
2	2	1 (READ)	06,07,08	129 (RESPONSE)	17,28
2	2			130 (UNSOL_RESP)	17,28
2	3	1 (READ)	06,07,08	129 (RESPONSE)	17,28
2	3			130 (UNSOL_RESP)	17,28

Table 104 —g3 double-bit binary input static objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
3	0	1 (READ)	00,01,06,17,28		
3	0	22 (ASSIGN_CLASS)	00,01,06,17,28		
3	1	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
3	2	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28

Table 105 —g4 double-bit binary input event objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
4	0	1 (READ)	06,07,08		
4	1	1 (READ)	06,07,08	129 (RESPONSE)	17,28
4	1			130 (UNSOL_RESP)	17,28
4	2	1 (READ)	06,07,08	129 (RESPONSE)	17,28
4	2			130 (UNSOL_RESP)	17,28
4	3	1 (READ)	06,07,08	129 (RESPONSE)	17,28
4	3			130 (UNSOL_RESP)	17,28

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 106 —g10 binary output static objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
10	0	1 (READ)	00,01,06,17,28		
10	0	22 (ASSIGN_CLASS)	00,01,06,17,28		
10	1	2 (WRITE)	00,01		
10	2	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28

Table 107 —g11 binary output event objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
11	0	1 (READ)	06,07,08		
11	1	1 (READ)	06,07,08	129 (RESPONSE)	17,28
11	1			130 (UNSOL_RESP)	17,28
11	2	1 (READ)	06,07,08	129 (RESPONSE)	17,28
11	2			130 (UNSOL_RESP)	17,28

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 108 —g12 binary output command objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
12	0	22 (ASSIGN_CLASS)	00,01,06,17,28		
12	1	3 (SELECT)	00,01,17,28	129 (RESPONSE)	Echo request
12	1	4 (OPERATE)	00,01,17,28	129 (RESPONSE)	Echo request
12	1	5 (DIRECT_OPERATE))	00,01,17,28	129 (RESPONSE)	Echo request
12	1	6 (DIRECT_OPERATE_NR)	00,01,17,28		
12	2	3 (SELECT)	07, 08 (Qty = 1)	129 (RESPONSE)	Echo request
12	2	4 (OPERATE)	07, 08 (Qty = 1)	129 (RESPONSE)	Echo request
12	2	5 (DIRECT_OPERATE)	07, 08 (Qty = 1)	129 (RESPONSE)	Echo request
12	2	6 (DIRECT_OPERATE_NR)	07, 08 (Qty = 1)		
12	3	3 (SELECT)	00 01	129 (RESPONSE)	Echo request
12	3	4 (OPERATE)	00,01	129 (RESPONSE)	Echo request
12	3	5 (DIRECT_OPERATE)	00,01	129 (RESPONSE)	Echo request
12	3	6 (DIRECT_OPERATE_NR)	00,01		

Table 109 —g11 binary output command event objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
13	0	1 (READ)	06,07,08		
13	1	1 (READ)	06,07,08	129 (RESPONSE)	17,28
13	1			130 (UNSOL_RESP)	17,28
13	2	1 (READ)	06,07,08	129 (RESPONSE)	17,28
13	2			130 (UNSOL_RESP)	17,28

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 110 —g20 counter static objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
20	0	1 (READ)	00,01,06,17,28		
20	0	7 (IMMED_FREEZE)	00,01,06,17,28		
20	0	8 (IMMED_FREEZE_NR)	00,01,06,17,28		
20	0	9 (FREEZE_CLEAR)	00,01,06,17,28		
20	0	10 (FREEZE_CLEAR_NR)	00,01,06,17,28		
20	0	11 (FREEZE_AT_TIME)	00,01,06,17,28		
20	0	22 (ASSIGN_CLASS)	00,01,06,17,28		
20	1	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
20	2	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
20	5	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
20	6	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28

Table 111 —g21 frozen counter static objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
21	0	1 (READ)	00,01,06,17,28		
21	0	22 (ASSIGN_CLASS)	00,01,06,17,28		
21	1	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
21	2	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
21	5	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
21	6	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
21	9	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
21	10	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 112 —g22 counter event objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
22	0	1 (READ)	06,07,08		
22	1	1 (READ)	06,07,08	129 (RESPONSE)	17,28
22	1			130 (UNSOL_RESP)	17,28
22	2	1 (READ)	06,07,08	129 (RESPONSE)	17,28
22	2			130 (UNSOL_RESP)	17,28
22	5	1 (READ)	06,07,08	129 (RESPONSE)	17,28
22	5			130 (UNSOL_RESP)	17,28
22	6	1 (READ)	06,07,08	129 (RESPONSE)	17,28
22	6			130 (UNSOL_RESP)	17,28

Table 113 —g23 frozen counter event objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
23	0	1 (READ)	06,07,08		
23	1	1 (READ)	06,07,08	129 (RESPONSE)	17,28
23	1			130 (UNSOL_RESP)	17,28
23	2	1 (READ)	06,07,08	129 (RESPONSE)	17,28
23	2			130 (UNSOL_RESP)	17,28
23	5	1 (READ)	06,07,08	129 (RESPONSE)	17,28
23	5			130 (UNSOL_RESP)	17,28
23	6	1 (READ)	06,07,08	129 (RESPONSE)	17,28
23	6			130 (UNSOL_RESP)	17,28

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 114 —g30 analog input static objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
30	0	1 (READ)	00,01,06,17,28		
30	0	7 (IMMED_FREEZE)	00,01,06,17,28		
30	0	8 (IMMED_FREEZE_NR)	00,01,06,17,28		
30	0	11 (FREEZE_AT_TIME)	00,01,06,17,28		
30	0	22 (ASSIGN_CLASS)	00,01,06,17,28		
30	1	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
30	2	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
30	3	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
30	4	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
30	5	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
30	6	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28

Table 115 — g31 frozen analog input static objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
31	0	1 (READ)	00,01,06,17,28		
31	0	22 (ASSIGN_CLASS)	00,01,06,17,28		
31	1	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
31	2	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
31	3	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
31	4	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
31	5	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
31	6	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
31	7	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
31	8	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 116 —g32 analog input event objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
22	0	1 (READ)	06,07,08	129 (RESPONSE)	17,28
22	1	1 (READ)	06,07,08	129 (RESPONSE)	17,28
22	1			130 (UNSOL_RESP)	17,28
22	2	1 (READ)	06,07,08	129 (RESPONSE)	17,28
22	2			130 (UNSOL_RESP)	17,28
22	3	1 (READ)	06,07,08	129 (RESPONSE)	17,28
22	3			130 (UNSOL_RESP)	17,28
32	6	1 (READ)	06,07,08	129 (RESPONSE)	17,28
32	6			130 (UNSOL_RESP)	17,28
22	5	1 (READ)	06,07,08	129 (RESPONSE)	17,28
22	5			130 (UNSOL_RESP)	17,28
22	7	1 (READ)	06,07,08	129 (RESPONSE)	17,28
22	7			130 (UNSOL_RESP)	17,28
32	8	1 (READ)	06,07,08	129 (RESPONSE)	17,28
32	8			130 (UNSOL_RESP)	17,28

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 117 —g33 frozen analog input event objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
33	0	1 (READ)	06,07,08		
33	1	1 (READ)	06,07,08	129 (RESPONSE)	17,28
33	1			130 (UNSOL_RESP)	17,28
33	2	1 (READ)	06,07,08	129 (RESPONSE)	17,28
33	2			130 (UNSOL_RESP)	17,28
33	3	1 (READ)	06,07,08	129 (RESPONSE)	17,28
33	3			130 (UNSOL_RESP)	17,28
33	4	1 (READ)	06,07,08	129 (RESPONSE)	17,28
33	4			130 (UNSOL_RESP)	17,28
33	5	1 (READ)	06,07,08	129 (RESPONSE)	17,28
33	5			130 (UNSOL_RESP)	17,28
33	6	1 (READ)	06,07,08	129 (RESPONSE)	17,28
33	6			130 (UNSOL_RESP)	17,28
33	7	1 (READ)	06,07,08	129 (RESPONSE)	17,28
33	7			130 (UNSOL_RESP)	17,28
33	8	1 (READ)	06,07,08	129 (RESPONSE)	17,28
33	8			130 (UNSOL_RESP)	17,28

Table 118 —g34 analog input reporting deadband objects

Group	Variation	Request (outstation shall parse)		Response (master shall parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
34	0	1 (READ)	00,01, 06		
34	1	1 (READ)	00,01, 06	129 (RESPONSE)	00,01
34	1	2 (WRITE)	00,01,06		
34	2	1 (READ)	00,01, 06	129 (RESPONSE)	00,01
34	2	2 (WRITE)	00,01,06		
34	3	1 (READ)	00,01, 06	129 (RESPONSE)	00,01
34	3	2 (WRITE)	00,01,06		

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 119 —g40 analog output status objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
40	0	1 (READ)	00,01, 06		
40	0	22 (ASSIGN_CLASS)	00,01,06,17,28		
40	1	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
40	2	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
40	3	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
40	4	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28

Table 120 —g41 analog output command objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
41	0	22 (ASSIGN_CLASS)	00,01,06,17,28		
41	1	3 (SELECT)	00,01,17,28	129 (RESPONSE)	Echo request
41	1	4 (OPERATE)	00,01,17,28	129 (RESPONSE)	Echo request
41	1	5 (DIRECT_OPERATE))	00,01,17,28	129 (RESPONSE)	Echo request
41	1	6 (DIRECT_OPERATE_NR)	00,01,17,28		
41	2	3 (SELECT)	00,01,17,28	129 (RESPONSE)	Echo request
41	2	4 (OPERATE)	00,01,17,28	129 (RESPONSE)	Echo request
41	2	5 (DIRECT_OPERATE))	00,01,17,28	129 (RESPONSE)	Echo request
41	2	6 (DIRECT_OPERATE_NR)	00,01,17,28		
41	3	3 (SELECT)	00,01,17,28	129 (RESPONSE)	Echo request
41	3	4 (OPERATE)	00,01,17,28	129 (RESPONSE)	Echo request
41	3	5 (DIRECT_OPERATE))	00,01,17,28	129 (RESPONSE)	Echo request
41	3	6 (DIRECT_OPERATE_NR)	00,01,17,28		
41	4	3 (SELECT)	00,01,17,28	129 (RESPONSE)	Echo request
41	4	4 (OPERATE)	00,01,17,28	129 (RESPONSE)	Echo request
41	4	5 (DIRECT_OPERATE))	00,01,17,28	129 (RESPONSE)	Echo request
41	4	6 (DIRECT_OPERATE_NR)	00,01,17,28		

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 121 —g42 analog output event objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
42	0	1 (READ)	06,07,08	129 (RESPONSE)	17,28
42	1	1 (READ)	06,07,08	129 (RESPONSE)	17,28
42	1			130 (UNSOL_RESP)	17,28
42	2	1 (READ)	06,07,08	129 (RESPONSE)	17,28
42	2			130 (UNSOL_RESP)	17,28
42	3	1 (READ)	06,07,08	129 (RESPONSE)	17,28
42	3			130 (UNSOL_RESP)	17,28
42	4	1 (READ)	06,07,08	129 (RESPONSE)	17,28
42	4			130 (UNSOL_RESP)	17,28
42	5	1 (READ)	06,07,08	129 (RESPONSE)	17,28
42	5			130 (UNSOL_RESP)	17,28
42	6	1 (READ)	06,07,08	129 (RESPONSE)	17,28
42	6			130 (UNSOL_RESP)	17,28
42	7	1 (READ)	06,07,08	129 (RESPONSE)	17,28
42	7			130 (UNSOL_RESP)	17,28
42	8	1 (READ)	06,07,08	129 (RESPONSE)	17,28
42	8			130 (UNSOL_RESP)	17,28

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 122 —g43 analog output command event objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
43	0	1 (READ)	06,07,08	129 (RESPONSE)	17,28
43	1	1 (READ)	06,07,08	129 (RESPONSE)	17,28
43	1			130 (UNSOL_RESP)	17,28
43	2	1 (READ)	06,07,08	129 (RESPONSE)	17,28
43	2			130 (UNSOL_RESP)	17,28
43	3	1 (READ)	06,07,08	129 (RESPONSE)	17,28
43	3			130 (UNSOL_RESP)	17,28
43	4	1 (READ)	06,07,08	129 (RESPONSE)	17,28
43	4			130 (UNSOL_RESP)	17,28
43	5	1 (READ)	06,07,08	129 (RESPONSE)	17,28
43	5			130 (UNSOL_RESP)	17,28
43	6	1 (READ)	06,07,08	129 (RESPONSE)	17,28
43	6			130 (UNSOL_RESP)	17,28
43	7	1 (READ)	06,07,08	129 (RESPONSE)	17,28
43	7			130 (UNSOL_RESP)	17,28
43	8	1 (READ)	06,07,08	129 (RESPONSE)	17,28
43	8			130 (UNSOL_RESP)	17,28

Table 123 —g50–g52 time information objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
50	1	1 (READ)	06	129 (RESPONSE)	07 (Qty = 1)
50	2	11 (FREEZE_AT_TIME)	07 (Qty = 1)		
50	2	12 (FREEZE_AT_TIME_NR)	07 (Qty = 1)		
50	3	2 (WRITE)	07 (Qty = 1)		
51	1	—	—	129 (RESPONSE)	07 (Qty = 1)
51	1			130 (UNSOL_RESP)	07 (Qty = 1)
51	2	—	—	129 (RESPONSE)	07 (Qty = 1)
51	2			130 (UNSOL_RESP)	07 (Qty = 1)
52	1	—	—	129 (RESPONSE)	07 (Qty = 1)
52	2	—	—	129 (RESPONSE)	07 (Qty = 1)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 124 —g70 file objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
70	2	29 (FILE_AUTHENTICATE)	5B (Cnt = 1)	129 (RESPONSE)	5B (Cnt = 1)
70	3	25 (OPEN_FILE)	5B (Cnt = 1)		
70	3	27 (DELETE_FILE)	5B (Cnt = 1)		
70	4	26 (CLOSE_FILE)	5B (Cnt = 1)	129 (RESPONSE)	5B (Cnt = 1)
70	4	30 (FILE_ABORT)	5B (Cnt = 1)	129 (RESPONSE)	5B (Cnt = 1)
70	4			130 (UNSOL_RESP)	5B (Cnt = 1)
70	5	1 (READ)	5B (Cnt = 1)	129 (RESPONSE)	5B (Cnt = 1)
70	5	2 (WRITE)	5B (Cnt = 1)	129 (RESPONSE)	5B (Cnt = 1)
70	5			130 (UNSOL_RESP)	5B (Cnt = 1)
70	6	1 (READ)	5B (Cnt = 1)	129 (RESPONSE)	5B (Cnt = 1)
70	6			130 (UNSOL_RESP)	5B (Cnt = 1)
70	7	28 (GET_FILE_INFO)	5B (Cnt = 1)	129 (RESPONSE)	5B (Cnt = 1)
70	7			130 (UNSOL_RESP)	5B (Cnt = 1)
70	8	31(ACTIVATE_CONFIG)	5B		

Table 125 —g80–g83 information objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
80	1	1 (READ)	00,01	129 (RESPONSE)	00,01
80	1	2 (WRITE)	00,01		
81	1	1 (READ)	06	129 (RESPONSE)	07
82	1	—	—	129 (RESPONSE)	5B (Cnt = 1)
82	1			130 (UNSOL_RESP)	5B (Cnt = 1)
83	1	1 (READ)	00,01,06	129 (RESPONSE)	5B
83	1			130 (UNSOL_RESP)	5B
83	2	1 (READ)	00,01,06	129 (RESPONSE)	5B

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 126 —g85–g88 data set objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
85	0	1 (READ)	06		
85	1	1 (READ)	00,01,06,17,28	129 (RESPONSE)	5B
85	1	2 (WRITE)	5B		
86	0	22 (ASSIGN CLASS)	00,01,06,17,28		
86	1	1 (READ)	00,01,06,17,28	129 (RESPONSE)	5B
86	1	2 (WRITE)	5B		
86	2	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
86	3	1 (READ)	00,01,06,17,28	129 (RESPONSE)	5B
86	3	2 (WRITE)	5B		
87	0	1 (READ)	06		
87	1	1 (READ)	00,01,06,17,28	129 (RESPONSE)	5B
87	1	2 (WRITE)	5B		
87	1	3 (SELECT)	5B	129 (RESPONSE)	5B
87	1	4 (OPERATE)	5B	129 (RESPONSE)	5B
87	1	5 (DIRECT_OPERATE)	5B	129 (RESPONSE)	5B
87	1	6 (DIRECT_OPERATE_NR)	5B		
88	0	1 (READ)	06,07,08		
88	1	1 (READ)	06,07,08	129 (RESPONSE)	5B
88	1			130 (UNSOL_RESP)	5B

Table 127 —g90 application & status of operation information objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
90	1	16 (INITIALIZE_APPLICATION)	5B		
90	1	17 (START_APPLICATION)	5B		
90	1	18 (STOP_APPLICATION)	5B		
91	1			129 (RESPONSE)	5B

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 128 —g101, g102 numeric static objects

Group	Variation	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
101	1	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
101	2	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
101	3	1 (READ)	00,01,06,17,28	129 (RESPONSE)	00,01,17,28
102	1	1 (READ)	00,01,03,04,05,06,17,28	129 (RESPONSE)	00,01, 03,04,05,17,28
102	1	2 (WRITE)	00,01,03,04,05,17,28		

Table 129 —g110–g113 string and virtual terminal static and event objects

Group	Variation (length)	Request (outstation may parse)		Response (master may parse)	
		Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
110	1–255	1 (READ)	00,01,03,04,05,06,17,28	129 (RESPONSE)	00,01, 03,04,05,17,28
110	1–255	2 (WRITE)	00,01,03,04,05,17,28	—	—
110	1–255	31 (ACTIVATE_CONFIG)	5B		
111	1–255	1 (READ)	06	129 (RESPONSE)	00,01, 03,04,05,17,28
111	1–255			130 (UNSOL_RESP)	00,01, 17,28
112	1–255	2 (WRITE)	00,01,17,28		
113	0	1 (READ)	00,01,17,28		
113	0	22 (ASSIGN_CLASS)	00,01,06,17,28		
113	1–255	—	—	129 (RESPONSE)	00,01, 17,28
113	1–255			130 (UNSOL_RESP)	00,01, 17,28

Table 130 —Function codes not used with objects

Request (outstation may parse)	Response (master may parse)
Function codes (decimal)	Function codes (decimal)
14 (WARM_RESTART)	
24 (RECORD CURRENT TIME)	

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

13. IP networking

13.1 IP networking overview

This specification defines the method for transporting DNP3 messages over the Internet Protocol suite.

13.1.1 IP networking purpose

DNP3 was originally designed for serial point-to-point SCADA communication (e.g., RS-232) with limited support for half-duplex serial networks (e.g., RS-485). Many users recognized a need for devices to exchange DNP3 messages over high-speed digital networks. This specification defines how to transport DNP3 traffic across a network using the Internet Protocol suite.

13.1.2 IP networking scope

This specification applies to all end devices that can interpret and generate DNP3 messages on a local (LAN) or wide area network (WAN). It does not cover devices that pass messages that may contain DNP3 as part of the network infrastructure (e.g., routers, switches, gateways, etc.) or devices that perform a physical conversion of the data (e.g., bridges, port servers, modems, etc.). This specification employs the Internet Protocol suite for message transport and does not redefine any of the existing DNP3 layers.

13.1.3 IP networking suite and device identification

The Internet Protocol suite is sometimes referred to as the TCP/IP protocol. For this document, the Internet Protocol suite is defined to include the protocols specified in IETF RFC 793 (TCP), IETF RFC 768 (UDP), and IETF RFC 791 (IP).

Throughout this document, the term “device” as it applies to a DNP3 device may be interpreted to be either a physical unit (e.g., a standalone product, RTU, IED, etc.) or a logical entity within a physical unit (e.g., logical RTU, virtual IED, etc.).

13.1.4 Protocol stack

The most attractive reasons for choosing the Internet Protocol suite as a transport mechanism for DNP3 are as follows:

— Seamless integration of the SCADA LAN to the corporate WAN

— Leverage existing equipment and standards

The Internet Protocol suite is platform independent and supported universally. It is highly scalable (LAN to WAN), and many quality implementations exist for both embedded and workstation operating systems. The growth of the Internet has fueled the large availability of equipment and has proved that the Internet Protocol suite is capable of transporting tremendous quantities and types of data.

The Internet Protocol suite and DNP3 use a layering paradigm; each piece of the protocol stack in one station logically communicates with the corresponding piece in the other station(s). It is therefore easy to

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

build DNP3 “on top of" the Internet Protocol suite since the Internet layers appear transparent to the DNP3 layers (see Figure 77).

Figure 77—Protocol stack

This clause specifies how to link the DNP3 layers with the Internet Protocol suite layers as well as the configuration parameters and methods to facilitate the network connection. The “Connection Management Layer” encapsulates this functionality.

13.2 Layer requirements

13.2.1 DNP3 link, transport, and application

The DNP3 Link, Transport, and Application Layers require no change of specification except in the time synchronization method for network operation. All of the requirements in the other protocol specification documents are applicable. Link frames are transported unchanged over the Internet Protocol suite under the control of the connection management layer.

13.2.1.1 Confirmations

Devices shall not transmit confirmed Data Link Layer frames (CONFIRMED_USER_DATA) when communicating via the Internet Protocol suite. Application Layer confirmation is the same when communicating over IP networks as over serial channels.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

13.2.1.2 Message transfer

Once the connection management layer has established a TCP connection or provided a socket for UDP datagrams, the DNP3 Link Layer may transfer frames as needed. The type of polling, either solicited or unsolicited, is independent from the connection establishment. For example, a typical case is for the master to establish a TCP connection with an outstation, perform a class 1,2,3,0 request (integrity poll), enable unsolicited messages, and operate in an unsolicited mode from then on.

13.2.2 Internet Protocol suite

All DNP3 network devices shall use the Internet Protocol suite. This suite is composed of several protocols that work together to provide data transport services. The interface between the connection management layer and the protocol suite is typically implemented with an Application Programming Interface (API) into the IP Transport Layer with the most common being a "sockets" interface (either Berkeley or Winsock derived).

13.2.2.1 Configuration requirements

The Internet Protocol suite is composed of two protocols at the Transport Layer. Each is designed for a specific type of data delivery depending on the characteristics of the network and the end devices. DNP3 network devices shall support the use of both the TCP and the UDP.

a) Master stations **shall** provide the ability to select the type of Internet Protocol connection to establish with an outstation. The selections **shall** be (1) a TCP initiating end point, (2) a datagram end point, or (3) a TCP dual end point.

b) Outstations **shall** provide the ability to select the type of Internet Protocol connection to establish with a master station. The selections **shall** be (1) a TCP listening end point, (2) a datagram end point, or (optionally) (3) a TCP dual end point.

NOTE—These selection requirements denote functional capabilities. The actual naming, style, and manner of the configuration options are device specific.

c) An outstation **may** provide multiple end points to support multiple master stations. A master station **should** provide multiple end points to support multiple outstations. See 13.2.3.5.

13.2.2.2 Registered port number

The port number 20000 has been registered with the IANA (Internet Assigned Numbers Authority) for use with DNP3. All devices shall support communications using this port number.

13.2.2.2.1 Optional port numbers

DNP3 network devices are free to use additional Internet Port numbers as long as the basic requirements for using the DNP3 port number are met. For example, an outstation manufacturer may wish to provide other TCP listening ports to support redundancy features or alternate communication end points in the device. Masters **shall** provide the ability to change the Internet destination port numbers and addresses used for establishing a connection with an outstation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

13.2.2.3 IP address assignment

This document does not specify the method for IP address assignment for DNP3 devices.

13.2.3 Connection management

The connection management layer provides the interface between the DNP3 protocol layers and the Internet Protocol suite. It is responsible for setting up and tearing down TCP connections, transmitting and accepting UDP datagrams, and passing the DNP3 frames between the channel and the DNP3 Data Link Layer.

13.2.3.1 TCP usage

TCP is the recommended transport protocol to use for most DNP3 network connections. It has facilities that substantially improve the reliability of the data transfer and is the best choice for wide area networks. TCP provides a serial data stream between devices for the DNP3 Link Layer and thus enables common integration with existing serial codes. Since TCP is a connection-oriented protocol, both devices shall participate in the defined synchronization scheme in order to establish the connection.

13.2.3.1.1 Initiating end point

Masters **shall** support a TCP initiating end point. A master in this mode initiates the connection by performing a TCP active open.

13.2.3.1.2 Listening end point

Outstations **shall** support a TCP listening end point. An outstation in this mode listens for a connection request by performing a TCP passive open.

13.2.3.1.3 Dual end point

Masters **shall** support and outstations **may** support a TCP dual end point. A dual end point is defined as a process that can both listen for connection requests and perform an active open on the channel if required. If the device does support a dual end point, it **shall** verify that only one connection is in place to another device at a time.

The quiescent state of both the master and the outstation is in listen mode. Either side may initiate the connection; the master to send controls, integrity polls, etc. and the outstation to report the initial null response and unsolicited data. Once the connection is established, it may or may not be kept open by both ends. Either end **may** choose to close the connection after data transfer is complete or on a device-specific timeout. Each side **shall** return to the listen mode after the connection is closed. If both sides simultaneously initiate the connection, the connection initiated by the master **shall** be maintained while the other **shall** be dropped (i.e., the master has priority on the creation of TCP connections).

See 13.5 for the UML statecharts describing dual end point operation. These statecharts are provided to give guidance when implementing dual end points and do not provide all of the details necessary for DNP3 message transfer.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

13.2.3.1.4 Basic requirements

a) Outstations **shall** support a listening end point. Outstations **may** support a dual end point. An outstation **should** provide the ability to validate connection requests from masters by IP address.

b) Master stations **shall** support both an initiating end point and a dual end point. A master operating as a dual end point **shall** provide the ability to validate connection requests from outstations by IP address.

c) Master stations **shall** be able to initiate a TCP connection with an outstation to port number 20000. Outstations **shall** be able to listen for connection requests on port 20000. Masters **shall** and outstations **should** provide the option to operate on other port numbers.

d) Outstations configured as dual end points **shall** be able to initiate a TCP connection with a master to port number 20000. Masters configured as dual end points **shall** be able to listen for connection requests on port 20000. Masters **shall** and outstations **should** provide the option to operate on other port numbers.

13.2.3.1.5 Configuration parameter guidance

a) If an outstation supports a dual end point, it shall provide a configuration option that determines the end point type. This option shall have these selections: (1) listening end point or (2) dual end point.

b) If an outstation supports a dual end point, it shall provide configuration parameters setting the IP address and port number of the master with which to make a connection. An outstation **should** provide a configuration parameter setting the IP address of the master to support connection validation.

c) Masters shall provide a configuration option that determines the end point type. This option shall have these selections: (1) initiating end point or (2) dual end point.

d) Depending on the type of multiple master support, outstations shall provide configuration parameters setting the IP address of the master or the listening port number.

e) Masters shall provide configuration parameters setting the IP address and port number for each outstation.

f) Masters shall provide a configuration option to set the DNP3 destination address for each logical device in an outstation. The DNP3 destination address should be unique for all logical devices within an outstation.

13.2.3.2 UDP usage

All network devices **shall** support the UDP transport protocol. UDP provides for DNP3 networking the ability to broadcast to multiple destinations on a subnet. UDP can also be used on a highly reliable network where the additional overhead of TCP reliability is not needed. Because of its connectionless design, UDP has a lower octet overhead than TCP. This characteristic is important for pay-per-byte networks such as Cellular Digital Packet Data.

NOTE—The following discussion shows how port 20000 is used to meet the minimum requirements of a device. Other port numbers may be supported in addition to 20000 as specified in 13.2.2.2.1. See 13.2.3.2.5 for more general cases.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UDP is used by DNP3 networking in the following cases:

Case A

When the communication channel uses TCP, an outstation **shall** provide a UDP end point at port 20000 to receive broadcast messages from the master station. This permits the master to send, for example, freeze or disable unsolicited commands to all logical devices within an outstation. Outstations do not respond to broadcast messages.

Case B

When the communication channel uses UDP only, an outstation **shall** provide a UDP end point at port 20000 (or optionally other port numbers) to receive all messages from the master station. Responses **shall** be returned to the master with the destination port number set to one of the following values:

— **B1:** 20000 (required to provide backward compatibility with version 1.0 implementations)

— **B2:** the source port number in the request (required to support)

— **B3:** a configured port number (optional)

Case C

Additionally, an outstation configured for unsolicited responses **shall** send its initial null response to either:

— **C1:** port 20000 (required)

— **C2:** a configured port number (optional)

If the initial null response is sent to 20000, the outstation **shall**, based on its configuration, use the source port number from subsequent requests or use 20000 to be backward compatible with version 1.0 masters. The recommended technique is to use the source port number from the first request and begin sending all responses, solicited and unsolicited, using this number as the destination port. The method of sending all UDP responses exclusively to port 20000 on the master station is deprecated and shall not be supported in the future.

If the initial unsolicited response is sent to a configured port number, the outstation **may**, based on its configuration, use the source port number from subsequent requests or continue sending all responses, solicited and unsolicited, using this configured number as the destination port. This method provides maximum control for a system designer to assign specific port numbers for individual outstations.

Figure 78 illustrates these cases:

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 78—UDP cases

NOTE—Cases A and B2 may provide the best options when network address translation (NAT) is used in system routers.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

13.2.3.2.1 Basic requirements

a) Masters **shall** support sending requests to UDP port 20000. Masters **shall** provide configuration parameters setting the IP address and port number for each outstation.

b) Outstations **shall** accept requests on UDP port 20000. Outstations **should** provide a configuration parameter setting the accepting port number for each master.

c) Masters **shall** set the source port of the request to the desired UDP destination port of the response.

d) Outstations **shall** be configurable to send responses with a destination port equal to the source port of the request.

e) To support compatibility with version 1.0 implementations, a master **shall** be configurable to accept responses on port 20000 and outstations **shall** be configurable to send responses to port 20000.

f) Outstations **shall** accept UDP datagrams on port 20000 when operating as a TCP listening or dual end point to facilitate broadcast messages from master stations. Outstations **should** only accept broadcast messages from masters whose IP address is associated with a valid TCP end point.

13.2.3.2.2 Configuration parameter guidance

a) An outstation shall be configurable to accept UDP requests only from specific master IP addresses or from any IP address.

b) An outstation shall provide a configuration option for each UDP end point that establishes how to set the destination UDP port number for responses. This option shall have a least these selections: (1) source port number from the request or (2) a configured port number. This option shall include the capability to enter or select 20000 for compatibility with version 1.0 masters. Figure 79 for shows an example of this configuration option.

c) An outstation may provide a configuration option for each UDP end point that specifies the UDP port number of the initial unsolicited response. If provided, this option shall be a configured port number, including the capability to enter or select 20000. The default configured port number shall be 20000. If this option is not provided, the initial unsolicited response shall be to port 20000.

d) A master shall provide user configuration for the outstation IP addresses from which it shall accept UDP responses. The master may reject responses from all other IP addresses. A master shall provide a configuration option for the destination port number on requests although 20000 shall be used in most cases.

e) A master station shall provide a configuration option for each UDP end point that establishes which port number shall be used to accept responses from the peer outstation. This option shall have at least these selections: (1) a port number chosen by the master when the request is sent or (2) a configured port number. This option shall include the capability to enter or select 20000 for compatibility with version 1.0 outstations. Figure 79 for shows an example of this configuration option.

f) A master station shall accept initial unsolicited responses on UDP port 20000 and may be configurable to accept them on a separately configured port number.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 79—Example configuration option for UDP port number

13.2.3.2.3 Multiple frames

A master station **shall** be capable of parsing multiple DNP3 Data Link Layer frames from a single UDP datagram. A DNP3 Data Link Layer frame **shall** not span multiple UDP datagrams.

13.2.3.2.4 Broadcast address

If UDP is used to broadcast a datagram to multiple devices on a LAN, the DNP3 address shall be set to one of the DNP3 broadcast addresses (65533 –65535) and all logical DNP3 devices at all IP addresses on the subnet shall process the encapsulated DNP3 Data Link Layer frame. Examples of broadcast events include freezes, enable/disable unsolicited, and time sync.

NOTE—Routers typically are not configured to forward broadcast messages. In practice, broadcast messages are only effective on a subnet.

13.2.3.2.5 Additional port numbers

Subclause 13.2.3.2 describes how port 20000 shall be supported to provide basic interoperability for all devices. As long as this minimum functionality is met, device manufacturers may support additional port numbers to facilitate more flexible system integration. For example, take cases A, B2, and C1 from Figure 78. These cases are redrawn in Figure 80 to show the use of a destination port number other than 20000.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 80—Example UDP cases with configurable port numbers

13.2.3.3 TCP connection status

If the listening end point of a TCP connection goes down (without purposely being disconnected by the software) and comes back up, the other end has no way of knowing until it sends a message. All DNP3 devices are required to provide a keep-alive mechanism as described in 13.2.3.3.1 so that each end of the connection can determine the on-line status of the other end when appropriate.

NOTE—TCP provides a keep-alive timer, but a typical implementation is 2 hours, which is not sufficient for the critical nature of the data carried by DNP3. Some Internet Protocol implementations allow this timeout to be changed, but this capability is not universal.

Outstations **shall** reject new connection requests from a master station that is already connected.

NOTE—An outstation may transmit a keep-alive message on the current connection when a new connection request is received in order to immediately determine the status of the current connection.

13.2.3.3.1 Keep-alive mechanism

All masters and outstations **shall** incorporate a keep-alive timer to monitor the status of an active TCP connection. This timer **shall** be restarted whenever any message is received from the other end. If the timer times out, the device **shall** transmit a keep-alive message. The keep-alive message **shall** be a DNP3 Data

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Link Layer status request (REQUEST_LINK_STATUS). If a response is not received to the keep-alive message, the connection **shall** be deemed broken and the appropriate action taken (see Figure 81).

Devices **shall** not explicitly disable transmission of keep-alive messages but may set the keep-alive interval sufficiently large to disable their transmission during periods of normal communication. As an example, in a polled environment, the time between normal polls for data may be shorter than the keep-alive interval. Each timely received poll request restarts the keep-alive timer, and as a result, the timer never times out. The device would only transmit keep-alive messages if polling were to cease.

NOTE—In a testing environment, it may be advantageous for a device to provide the ability to effectively disable the keep-alive mechanism so that the connection is maintained when the message traffic is intentionally very low. An example would be the case where a test set is used to send specific manually generated requests and to verify the responses. In this case, testing may be simplified if the master and outstation disable their keep-alive timers or provide the ability to set the timers to a very high interval. This mode is not required nor does it nullify the keep-alive requirement described in the preceding discussion.

Keep-alive messages are only needed when the connection is open.

If the keep-alive timeout is configurable, the vendor should publish the default keep-alive timer value as well as the range in the Device Profile Document.

13.2.3.3.2 Broken connections

Figure 81 describes the handling of broken TCP connections.

	Device	Symptom	Action
1	master	Loss of power or restart	Issue new connection request.
2	outstation	Loss of power or restart	Either create new listening end point or initiate connection to generate unsolicited response if configured as a dual end point.
3	master	Failure to receive a response to a keep-alive message	Close connection and issue new connection request.
4	outstation	Failure to receive a response to a keep-alive message	Configured as: Listening end point—close connection and commence listening for new connections. Dual end point—close connection, commence listening for new connections, and wait for need to connect.
5	master	Receive TCP RST flag indicating that the outstation has reset the connection	Close connection and issue new connection request.
6	outstation	Receive TCP RST flag indicating that the master has reset the connection	Configured as: Listening end point—close connection and commence listening for new connections. Dual end point—close connection, commence listening for new connections, and wait for need to connect.

Figure 81—Handling broken TCP connections

13.2.3.3.3 Closed connections

Figure 82 describes the handling of closed TCP connections.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

	Device	Symptom	Action
1	master	Receive TCP FIN flag indicating that the outstation has closed the connection	Close connection and issue new connection request.
2	outstation	Receive TCP FIN flag indicating that the master has closed the connection	Configured as: Listening end point—close connection and commence listening for new connections. Dual end point—close connection, commence listening for new connections, and wait for need to connect.

Figure 82—Handling closed TCP connections

13.2.3.4 Single master connection

Figure 83 depicts the TCP connection between a single master and outstation with the master configured as the initiating end point and the outstation as the listening end point. Alternatively, both devices may be configured as dual end points if the outstation supports it.

Figure 83—Single master connection

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

In this setup, the outstation listens on port 20000 and the master makes a connection request to the IP address of the outstation at port 20000. An outstation may support more than one logical DNP3 device; in which case, each logical device is accessed by a unique DNP3 destination address. The master shall have a corresponding configuration table for each logical device. The outstation shall maintain a set of state information (i.e., an association) for each logical device.

13.2.3.4.1 Requirements

All masters and outstations **shall** support a single master connection at port 20000.

13.2.3.5 Multiple master connections

In some systems, it is desirable to have multiple DNP3 master stations simultaneously accessing the same outstation. This may be required for some redundant schemes or if there are multiple users of the outstation data. Such a setup can require that the outstation have some manner for uniquely identifying the various masters depending on how the data needs to be reported. Each master may need access to different data or have different access permissions. The outstation shall maintain a set of state information that may consist of variables and event buffers for each master with which it communicates.

There are a few different methods for establishing connections between multiple masters and an outstation. The following subclauses define the methods applicable to DNP3. All methods require a single physical network connection and allow, but do not require, an outstation to have multiple DNP3 logical devices within the physical device.

13.2.3.5.1 Connection establishment—method 1

Connection based on master IP address (Figure 84).

Figure 84—Connection based on master IP address

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

This method requires that each collection of logical DNP3 devices has a unique and defined address of the DNP3 master. When a connection request is accepted, the connection management layer assigns the socket to the collection with the matching source IP address and proceeds to route all DNP3 messages from that master to this collection of logical devices. The individual logical devices are accessed in the collection based on the DNP3 destination address.

13.2.3.5.2 Connection establishment—method 2

Connection based on IP port number (Figure 85).

Figure 85—Connection based on port number

This method requires the connection management layer to listen on different TCP ports. Each collection of logical DNP3 devices is assigned to a specific port so the DNP3 message routing is predefined. The individual logical devices are accessed in the collection based on the DNP3 destination address.

13.2.3.5.3 Connection establishment—method 3

All connections accepted for browsing static data (Figure 86).

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 86—All connections accepted for browsing static data

This method is useful for allowing multiple masters to connect with the outstation to "browse" for static data. The outstation allows connections from any master. The outstation simply responds to requests for data based on the DNP3 destination address.

The outstation may be configured to limit the IP addresses that can establish a connection. It is also possible to support events by mapping, through configuration, the logical DNP3 devices to the DNP3 destination address and requiring each master to only access specific logical devices.

13.2.3.5.4 Requirements

Outstations are **not** required to support multiple master connections.

If an outstation supports multiple master connections, it **shall** provide connection establishment using method 1 outlined in 13.2.3.5.1.

If an outstation supports multiple master connections, it **should** provide connection establishment using method 2 outlined in 13.2.3.5.2.

If an outstation supports multiple master connections, it **may** provide connection establishment using method 3 outlined in 13.2.3.5.3.

13.2.3.6 Multiple outstation connections

In most systems, it is desirable to have multiple DNP3 outstations simultaneously accessible from a master station. The master shall be configurable to support the characteristics of the utility network so that it can make a connection with all outstations. Figure 87 depicts TCP connections between a single master with a single physical network connection and multiple outstations. The master is configured as the initiating end point. Alternatively, both ends of any of the connections may be configured as dual end points if the outstation supports it.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 87—Multiple outstation connections

13.2.3.6.1 Requirements

Master stations **should** support multiple outstation connections.

If a master supports multiple outstation connections, it **shall** provide the ability to configure the destination IP address of the outstation and the destination port number of the outstation.

13.3 Security

13.3.1 Rudimentary

All devices **should** be able to limit connection establishment based on IP address, address range, or validation features provided by higher level security layers. This recommendation is designed to protect against incorrectly configured systems where multiple devices attempting connections to the same device may cause data loss or inadvertent commands.

13.3.2 Advanced

Refer to Clause 7.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

13.3.3 External

Virtual private network (aka VPN) routers and secure Internet Protocols (e.g., IPSec) are recommended to provide site-to-site security, in accordance with the security policies of the organization.

13.4 Time synchronization

The source of the time on the network is system dependent and may be a master station or a dedicated time server. Time can be distributed on the network outside of DNP3 using various standardized methods. It can also be distributed, with some limitations, using specific DNP3 messages.

See Clause 10 for the DN3 network time synchronization method.

13.5 UML statecharts

13.5.1 Dual end point—master

Figure 88 diagrams the master station statechart for dual end point.

Figure 88—Master station statechart for dual end point

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

13.5.2 Dual end point—outstation

Figure 89 diagrams the outstation statechart for dual end point.

Figure 89—Outstation statechart for dual end point

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

14. Interoperability

14.1 About this clause

14.1.1 Purpose of this clause

This clause presents a standard template for describing configuration parameters associated with a DNP3 device and defines the implementation subsets, or levels, for the DNP3 Application Layer. It is intended to be used as a reference for determining compatibility between implementations of the DNP3 Application Layer.

This clause describes several subsets of the protocol, for the purpose of giving vendors and customers a common set of terms for describing what they have implemented and what they require, respectively.

14.1.2 Who should use this clause

This clause is intended to be used by the following people, although its audience is not necessarily limited to those listed here:

— Designers responsible for determining what portion of the protocol to implement.

— Systems engineers responsible for determining compatibility between components of a network.

— Purchasing agents or certification agencies responsible for determining whether a vendor's product meets customers' standards.

14.1.3 How this clause is organized

The organization of this clause is described as follows:

4.2 OVERVIEW describes basic concepts concerning DNP3 subsets: basic terminology, how to interpret the implementation tables, and the duties of devices implementing the subsets. It briefly discusses the goals used to develop the subsets.

4.3 LEVEL 1 DNP3 IMPLEMENTATION (DNP3-L1) describes the minimum subset of the protocol that can be implemented, typically between a master station and an IED.

4.4 LEVEL 2 DNP3 IMPLEMENTATION (DNP3-L2) describes a subset of the protocol, slightly larger than Level 1. It is typically implemented between a master station and a large IED or small RTU.

4.5 LEVEL 3 DNP3 IMPLEMENTATION (DNP3-L3) describes a subset of the protocol, larger than Level 2, that can be implemented between a master station and a more advanced RTU.

4.6 LEVEL 4 DNP3 IMPLEMENTATION (DNP3-L4) describes a subset of the protocol, larger than Level 3, incorporating data types and function codes that were added to DNP3 after definition of the first three original subset levels.

4.7 CONFORMANCE describes the conditions under which a device is said to conform to a given DNP3 subset.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

4.8 XML REPRESENTATION describes how the Device Profile Document is represented in a machine-readable format to allow for more efficient and accurate device setup. It also allows mapping DNP3 Data Points to IEC 61850 Object Models.

4.9 INSTRUCTIONS FOR CREATING A DEVICE PROFILE DOCUMENT (the Device Profile: Appendix 1 of Volume 8 of the DNP3 Specification [B1]) describes the information a device vendor shall include in a document describing a device's implementation of DNP3.

14.2 Overview

This subclause describes various concepts concerning DNP3 protocol subsets in general.

14.2.1 Terminology

This subclause defines some of the terms used throughout this clause.

When a master or outstation satisfies all the requirements of a particular DNP3 subset, it is said to implement a particular level of the protocol. The term "Level" is chosen so as to not conflict with DNP3 data classes or the OSI concept of layers. The abbreviation for a DNP3 subset implementation consists of "DNP3," a dash, and "L" followed by the level number.

For example, a vendor may be able to say, "This device implements the DNP3 Application Layer protocol Level 1," or just "This device implements DNP3 - L1."

NOTE— This clause specifies only the *minimum* requirements for a particular implementation level. A device may implement extra features in addition to these requirements and still conform. Refer to 14.7 for more details.

14.2.2 Reading the subset tables

The subsets of the DNP3 protocol shown in the Device Profile Document are described in a common format. Each subclause describing an implementation level contains a table having the following columns defining the function and qualifier codes used for both request and response messages:

Request column	These fields describe the set of DNP3 objects, function codes, and qualifiers an outstation shall be able to *parse* as a part of an incoming request in order to have implemented the subset. Function Codes A list of the request function codes that the outstation shall accept as operators on this object. Qualifier Codes A list of the qualifier codes the outstation shall parse in association with this object. If these fields are blank, it means the outstation does not need not be able to parse the specified object in order to implement the subset.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Response column	These fields describe the set of DNP3 objects, function codes, and qualifiers a master device shall be able to *parse* from an incoming response or unsolicited response in order to have implemented the subset. These fields also define the minimum subset of responses an outstation may make. Function Codes A list of the response function codes the master shall accept as operators on this object. Qualifier Codes A list of the qualifier codes the master shall parse in association with this object. If these fields are blank, it means the master does not need not be able to parse the specified object in order to implement the subset.

There are four levels of implementation of DNP3 defined within this document. Each level builds on the level preceding it.

NOTE—The subset describes the minimum set of objects, function codes, and qualifiers the devices should *parse* in order to implement the subset. If an object is supported, it should be used with the function and qualifier codes in order to implement the subset. It *does not* specify that an outstation should actually report inputs, outputs, and data for all of the objects in the subset.

For example, suppose a subset table contains the following line:

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	**Var num**	**Description**	**Function codes (dec)**	**Qualifier codes (hex)**	**Function codes (dec)**	**Qualifier codes (hex)**
30	4	Analog Input—16-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range,or all)	129 (response)	00, 01 (start-stop)
		. . . . .				

This entry indicates the following:

— An outstation shall accept and respond to any **read** request (Function Code 1) of 16-Bit Analog Input without flag objects. The **read** request may use either 8-bit start and stop indices (Qualifier Code 0x00), 16-bit start and stop indices (Qualifier Code 0x01), or no range field (Qualifier Code 0x06). The outstation need not parse **read** requests for 16-Bit Analog Input without flag objects using any other Qualifier Codes. The master device cannot include a 16-Bit Analog Input without flag object in any request containing a function code other than **read**.

— The outstation need not actually report any analog inputs. It may set an Internal Indication bit in its response, or return a null response, as discussed in Clause 4 through Clause 6.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— A master device shall be able to parse a **response** (Function Code 129) from the outstation containing 16-Bit Analog Input without flag objects. The response shall use 8-bit start and stop indices (Qualifier Code 0x00) or 16-bit start and stop indices (Qualifier Code 0x01). The master device need not parse responses or unsolicited responses containing 16-Bit Analog Input without flag objects using any other Qualifier Codes.

— The master need not use the data supplied by the outstation for any purpose but can discard it after sending any necessary confirmation.

14.2.3 Goals and assumptions

These subset definitions were prepared with the following goals:

— Minimize the complexity of implementation. Where complexity must be added, it was added at the master rather than at the outstation.

— Minimize the bandwidth used.

— Allow all data provided by an outstation to be retrieved at any level.

— Allow flexibility for a variety of different, but interoperable, implementations.

It was assumed that the simpler the outstation, the fewer points it would provide. Therefore, there would be less need for the more complex point range specifications available in DNP3.

As a result of these goals and assumptions, the lower level subset definitions are based around polls of Class Data Objects. In a typical minimum implementation, it is expected that a master device will poll frequently for class 1, 2, or 3 data, interspersed with infrequent class 0 data polls. More advanced implementations may take advantage of unsolicited responses and the more complex polling features available in DNP3.

Qualifiers are a complex part of DNP3, so the number of qualifiers required to be supported by either a master or an outstation was limited. Table 131 illustrates the subset of qualifiers chosen and the intended use of each qualifier. There may be a few exceptions to these rules.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 131 —Qualifiers used in the subset definitions

Qualifier (hex)	Use in a request	Use in a response
00, 01	A range of static points, or a single point with a point number	Static objects
06	All points	Not valid
07, 08	A limited quantity of events A single point with no number (e.g., Time and Date)	A single point with no number (e.g., Time and Date)
17, 28	Controls (usually one or more unrelated points)	Event objects (usually one or more unrelated points)

14.3 Level 1 DNP3 implementation (DNP3-L1)

This subclause describes the minimum subset of the DNP3 Application Layer that can be supported and still be said to implement the protocol. This implementation level is called Level 1 (L1).

14.3.1 Intended use

This level of implementation is intended to represent the simplest implementation of DNP3 for communicating between a master and a typical IED. It would typically be used between a master station or data concentrator and a small end device (e.g., meter, relay, auto-recloser, or capacitor bank controller). It is intended for use with an outstation whose input and output points are local to the device.

14.3.2 General description

The Level 1 subset is based around Class Data polling, as described in 14.2.3. A Level 1 outstation shall accept requests for:

— READs of Class Data Objects

— READs of Binary Output and Analog Output objects, if such outputs exist on the outstation.

NOTE—If such objects do not exist, the outstation is allowed to respond OBJECT UNKNOWN.

— READs of specific variations of group 0 (Device Attributes) objects

— Control operations to Binary Outputs or Analog Outputs, if they exist on the outstation. If such objects do not exist, the outstation is allowed to respond OBJECT UNKNOWN.

— WRITEs to the RESTART Internal Indication

— COLD RESTARTs

— DELAY MEASUREMENTs and WRITEs to Time and Date, if the outstation sets the TIME SYNCHRONIZATION REQUIRED Internal Indication (see “Time Synchronization” in the Device Profile: Appendix 1 of Volume 8 of the DNP3 Specification [B1])

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

The device shall provide an XML version of the Device Profile Document containing both the capabilities and the current values set in the device. This file may be obtained by means other than DNP3 File Transfer, such as generated by PC-based configuration software.

A Level 1 master shall accept a subset of object variations that is approximately one third of the total number defined in DNP3 but includes most basic data types:

— Binary Inputs and Events

— Counters and Counter Events

— Analog Inputs and Events

— Binary and Analog Output Status

The following are specifically NOT included:

— Frozen objects

— Time-stamped objects, with the exception of Binary Inputs because Sequence of Events information is a common industry requirement

NOTE—Although a Level 1 master shall be able to parse many objects, a Level 1 outstation is not *required* to generate them. For example, an outstation need not provide Binary Output Status or Analog Output Status objects if it does not control any of these outputs. The outstation may also choose to report Binary Input Without Time objects instead of Binary Input Change With Relative Time objects.

A Level 1 outstation may send unsolicited responses of some objects, but this capability shall be configurable. See "Unsolicited Responses" in the Device Profile: Appendix 1 of Volume 8 of the DNP3 Specification [B1].

Although not required, an outstation device should be tested to the applicable subset definition using the current IED certification procedures available.[38] As these procedures are updated from time to time (as often as annually), it is suggested that the reader make a practice of checking for updates to the IED certification procedures. Equipment purchasers are strongly encouraged to specify a requirement for equipment certification.

14.3.3 Implementation table

Table 132 describes the objects, function codes, and qualifiers used in a Level 1 DNP3 implementation.

[38] See http://www.dnp.org.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 132 —Level 1 implementation (DNP3-L1)

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
0	242	Device Attributes—Device manufacturer's software version	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	243	Device Attributes—Device manufacturer's hardware version	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	246	Device Attributes—User assigned ID code/number	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	248	Device Attributes—Device serial number	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	250	Device Attributes—Device manufacturer's product name and model	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	252	Device Attributes—Device manufacturer's name	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	254	Device Attributes—Non-specific all attributes request	1 (read)	00 (start-stop) 06 (no range, or all)		
0	255	Device Attributes—List of attribute variations	1 (read)	00 (start-stop) 06 (no range, or all)	129 (response)	00 (start-stop) 5B (free format)
1	1	Binary Input—Packed format			129 (response)	00, 01 (start-stop)
1	2	Binary Input—With flags			129 (response)	00, 01 (start-stop)
2	1	Binary Input Event—Without time			129 (response) 130 (unsol. resp)	17, 28 (index)
2	2	Binary Input Event—With absolute time			129 (response) 130 (unsol. resp)	17, 28 (index)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
2	3	Binary Input Event—With relative time			129 (response) 130 (unsol. resp)	17, 28 (index)
10	0	Binary Output—Any Variation	1 (read)	06 (no range,or all)		
10	2	Binary Output—Output status with flags			129 (response)	00, 01 (start-stop)
12	1	Binary Command—CROB	3 (select) 4 (operate) 5 (direct op) 6 (dir. op, no ack)	17, 28 (index)	129 (response)	echo of request
20	1	Counter—32-bit with flag			129 (response)	00, 01 (start-stop)
20	2	Counter—16-bit with flag			129 (response)	00, 01 (start-stop)
20	5	Counter—32-bit without flag			129 (response)	00, 01 (start-stop)
20	6	Counter—16-bit without flag			129 (response)	00, 01 (start-stop)
22	1	Counter Event—32-bit with flag			129 (response) 130 (unsol. resp)	17, 28 (index)
22	2	Counter Event—16-bit with flag			129 (response) 130 (unsol. resp)	17, 28 (index)
30	1	Analog Input—32-bit with flag			129 (response)	00, 01 (start-stop)
30	2	Analog Input—16-bit with flag			129 (response)	00, 01 (start-stop)
30	3	Analog Input—32-bit without flag			129 (response)	00, 01 (start-stop)
30	4	Analog Input—16-bit without flag			129 (response)	00, 01 (start-stop)
32	1	Analog Input Event—32-bit without time			129 (response) 130 (unsol. resp)	17, 28 (index)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
32	2	Analog Input Event—16-bit without time			129 (response) 130 (unsol. resp)	17, 28 (index)
40	0	Analog Output Status—Any Variation	1 (read)	06 (no range, or all)		
40	2	Analog Output Status—16-bit with flag			129 (response)	00, 01 (start-stop)
41	2	Analog Output—16-bit	3 (select) 4 (operate) 5 (direct op) 6 (dir. op, noack)	17, 28 (index)	129 (response)	echo of request
50	1	Time and Date—Absolute time	2 (write)	07 (limited qty = 1)		
51	1	Time and Date CTO—Absolute time, synchronized			129 (response) 130 (unsol. resp)	07 (limited qty) (qty = 1)
51	2	Time and Date CTO—Absolute time, unsynchronized			129 (response) 130 (unsol. resp)	07 (limited qty) (qty = 1)
52	1	Time Delay—Coarse			129 (response)	07 (limited qty) (qty = 1)
52	2	Time Delay—Fine			129 (response)	07 (limited qty) (qty = 1)
60	1	Class Objects—Class 0 data	1 (read)	06 (no range, or all)		
60	2	Class Objects—Class 1 data	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
60	3	Class Objects—Class 2 data	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
60	4	Class Objects—Class 3 data	1 (read)	06 (no range, or all) 07, 08 (limited qty)		

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
80	1	Internal Indications—Packed format	2 (write)	00 (start-stop) index=7		
No Object (function code only)			13 (cold restart)			
No Object (function code only)			23 (delay meas.)			

14.4 Level 2 DNP3 implementation (DNP3-L2)

This subclause describes the second smallest subset of the DNP3 Application Layer. This implementation level is called Level 2 (L2).

14.4.1 Intended use

This level contains a few more features than the Level 1 implementation. It is intended for communications between a master station or data concentrator and a device that could be called either a large IED or a small RTU. Typically, the input and output points of such a device would be local to the device.

14.4.2 General description

A Level 2 outstation implementation is the same as a Level 1 outstation implementation with the following additions:

— A Level 2 outstation accepts FREEZE requests on Binary Counter objects (not Analog Input objects or Frozen Counters). See “Freeze Operations” in the Device Profile: Appendix 1 of Volume 8 of the DNP3 Specification [B1].

— A Level 2 outstation parses READ requests for variation 0 of specific objects.

— A Level 2 outstation parses READ requests for variations 1, 2, and 3 of Binary Input Change objects.

— A Level 2 outstation parses READ requests for Frozen Counter objects and may report Frozen Counter objects (but not Frozen Delta Counters).

14.4.3 Implementation table

Table 133 describes the objects, function codes, and qualifiers used in a Level 2 DNP3 implementation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 133 —Level 2 implementation (DNP3-L2)

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
0	242	Device Attributes—Device manufacturer's software version	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	243	Device Attributes—Device manufacturer's hardware version	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	246	Device Attributes—User assigned ID code/number	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	248	Device Attributes—Device serial number	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	250	Device Attributes—Device manufacturer's product name and model	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	252	Device Attributes—Device manufacturer's name	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	254	Device Attributes—Non-specific all attributes request	1 (read)	00 (start-stop) 06 (no range, or all)		
0	255	Device Attributes—List of attribute variations	1 (read)	00 (start-stop) 06 (no range, or all)	129 (response)	00 (start-stop) 5B (free format)
1	0	Binary Input—Any Variation	1 (read)	06 (no range,or all)		
1	1	Binary Input—Packed format			129 (response)	00, 01 (start-stop)
1	2	Binary Input—With flags			129 (response)	00, 01 (start-stop)
2	0	Binary Input Event—Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
2	1	Binary Input Event—Without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
2	2	Binary Input Event—With absolute time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
2	3	Binary Input Event—With relative time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
10	0	Binary Output—Any Variation	1 (read)	06 (no range,or all)		
10	2	Binary Output—Output status with flags			129 (response)	00, 01 (start-stop)
12	1	Binary Command –CROB	3 (select) 4 (operate) 5 (direct op) 6 (dir. op, no ack)	17, 28 (index)	129 (response)	echo of request
20	0	Counter—Any Variation	1 (read) 7 (freeze) 8 (freeze noack) 9 (freeze clear) 10 (frz. cl. noack)	06 (no range,or all)		
20	1	Counter—32-bit with flag			129 (response)	00, 01 (start-stop)
20	2	Counter—16-bit with flag			129 (response)	00, 01 (start-stop)
20	5	Counter—32-bit without flag			129 (response)	00, 01 (start-stop)
20	6	Counter—16-bit without flag			129 (response)	00, 01 (start-stop)
21	0	Frozen Counter—Any Variation	1 (read)	06 (no range,or all)		
21	1	Frozen Counter—32-bit with flag			129 (response)	00, 01 (start-stop)
21	2	Frozen Counter—16-bit with flag			129 (response)	00, 01 (start-stop)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
21	9	Frozen Counter—32-bit without flag			129 (response)	00, 01 (start-stop)
21	10	Frozen Counter—16-bit without flag			129 (response)	00, 01 (start-stop)
22	0	Counter Event—Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
22	1	Counter Event—32-bit with flag			129 (response) 130 (unsol. resp)	17, 28 (index)
22	2	Counter Event—16-bit with flag			129 (response) 130 (unsol. resp)	17, 28 (index)
30	0	Analog Input—Any Variation	1 (read)	06 (no range,or all)		
30	1	Analog Input—32-bit with flag			129 (response)	00, 01 (start-stop)
30	2	Analog Input—16-bit with flag			129 (response)	00, 01 (start-stop)
30	3	Analog Input—32-bit without flag			129 (response)	00, 01 (start-stop)
30	4	Analog Input—16-bit without flag			129 (response)	00, 01 (start-stop)
32	0	Analog Input Event—Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
32	1	Analog Input Event—32-bit without time			129 (response) 130 (unsol. resp)	17, 28 (index)
32	2	Analog Input Event—16-bit without time			129 (response) 130 (unsol. resp)	17, 28 (index)
40	0	Analog Output Status—Any Variation	1 (read)	06 (no range,or all)		
40	2	Analog Output Status—16-bit with flag			129 (response)	00, 01 (start-stop)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
41	2	Analog Output—16-bit	3 (select) 4 (operate) 5 (direct op) 6 (dir. op, no ack)	17, 28 (index)	129 (response)	echo of request
50	1	Time and Date—Absolute time	2 (write)	07 (limited qty = 1)		
51	1	Time and Date CTO—Absolute time, synchronized			129 (response) 130 (unsol. resp)	07 (limited qty) (qty = 1)
51	2	Time and Date CTO—Absolute time, unsynchronized			129 (response) 130 (unsol. resp)	07 (limited qty) (qty = 1)
52	1	Time Delay—Coarse			129 (response)	07 (limited qty) (qty = 1)
52	2	Time Delay—Fine			129 (response)	07 (limited qty) (qty = 1)
60	1	Class Objects—Class 0 data	1 (read)	06 (no range,or all)		
60	2	Class Objects—Class 1 data	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
60	3	Class Objects—Class 2 data	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
60	4	Class Objects—Class 3 data	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
80	1	Internal Indications—Packed format	2 (write)	00 (start-stop) index=7		
No Object (function code only)			13 (cold restart)			
No Object (function code only)			23 (delay meas.)			

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

14.5 Level 3 DNP3 implementation (DNP3-L3)

This subclause describes a subset of the DNP3 Application Layer that contains more functionality than the Level 2 subset.

14.5.1 Intended use

This level of implementation of DNP3 is for communicating between a master and a medium-size outstation device (e.g., RTU or Data Concentrator).

14.5.2 General description

A Level 3 implementation uses a larger range of objects, variations, functions, and qualifier codes than a Level 2 implementation.

A Level 3 outstation implementation is the same as a Level 2 implementation with the following additions:

— A Level 3 outstation shall process read requests for many specific objects and variations.

— A Level 3 outstation shall process read requests for all group 0 (Device Attribute) variations

— A Level 3 outstation shall process write requests for specific variations of group 0 (Device Attributes) objects

— A Level 3 outstation shall process a larger range of requests with a larger range of function codes.

— A Level 3 implementation supports enabling and disabling unsolicited responses on a class-by-class basis. A master can enable or disable unsolicited responses for class 1, class 2, and class 3 objects only. The request fragment may contain one or more of the following object headers only (refer to Clause 4 through Clause 6):

  1) Class 1 (group 60, variation 2, qualifier 06)
  2) Class 2 (group 60, variation 3, qualifier 06)
  3) Class 3 (group 60, variation 4, qualifier 06)

— A Level 3 implementation supports the assigning and re-assigning of data objects to classes dynamically (i.e., during runtime). An **assign class** (Function Code 22) request shall contain a class 0, 1, 2, or 3 object header followed by one or more static data object headers. Several sets of class and data object headers may be contained in a single request fragment (refer to Clause 4 through Clause 6).

14.5.3 Implementation table

Table 134 describes the objects, function codes, and qualifiers used in a Level 3 DNP3 implementation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table 134 —Level 3 implementation (DNP3-L3)

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
0	211	Device Attributes—Identifier of support for user-specific attributes	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	212	Device Attributes—Number of master-defined data set prototypes	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	213	Device Attributes—Number of outstation-defined data set prototypes	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	214	Device Attributes—Number of master-defined data sets	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	215	Device Attributes—Number of outstation-defined data sets	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	216	Device Attributes—Max number of binary outputs per request	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	217	Device Attributes—Local timing accuracy	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	218	Device Attributes—Duration of timing accuracy	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	219	Device Attributes—Support for analog output events	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	220	Device Attributes—Max analog output index	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	221	Device Attributes—Number of analog outputs	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
0	222	Device Attributes—Support for binary output events	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	223	Device Attributes—Max binary output index	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	224	Device Attributes—Number of binary outputs	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	225	Device Attributes—Support for frozen counter events	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	226	Device Attributes—Support for frozen counters	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	227	Device Attributes—Support for counter events	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	228	Device Attributes—Max counter index	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	229	Device Attributes—Number of counter points	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	230	Device Attributes—Support for frozen analog inputs	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	231	Device Attributes—Support for analog input events	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	232	Device Attributes—Maximum analog input index	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	233	Device Attributes—Number of analog input points	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
0	234	Device Attributes—Support for double-bit binary input events	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	235	Device Attributes—Maximum double-bit binary input index	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	236	Device Attributes—Number of double-bit binary input points	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	237	Device Attributes—Support for binary input events	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	238	Device Attributes—Max binary input index	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	239	Device Attributes—Number of binary input points	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	240	Device Attributes—Max transmit fragment size	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
			2 (write)	00 (start-stop)		
0	241	Device Attributes—Max receive fragment size	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	242	Device Attributes—Device manufacturer's software version	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	243	Device Attributes—Device manufacturer's hardware version	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	245	Device Attributes—User-assigned location name	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
			2 (write)	00 (start-stop)		

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
0	246	Device Attributes—User assigned ID code/number	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
			2 (write)	00 (start-stop)		
0	247	Device Attributes—User-assigned device name	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
			2 (write)	00 (start-stop)		
0	248	Device Attributes—Device serial number	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	249	Device Attributes—DNP3 subset and conformance	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	250	Device Attributes—Device manufacturer's product name and model	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	252	Device Attributes—Device manufacturer's name	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	254	Device Attributes—Non-specific all attributes request	1 (read)	00 (start-stop) 06 (no range, or all)		
0	255	Device Attributes—List of attribute variations	1 (read)	00 (start-stop) 06 (no range, or all)	129 (response)	00 (start-stop) 5B (free format)
1	0	Binary Input—Any Variation	1 (read) 22 (assign class)	00, 01 (start-stop) 06 (no range, or all)		
1	1	Binary Input—Packed format	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
1	2	Binary Input—With flags	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
2	0	Binary Input Event—Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
2	1	Binary Input Event—Without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
2	2	Binary Input Event—With absolute time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
2	3	Binary Input Event—With relative time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
10	0	Binary Output—Any Variation	1 (read)	00, 01 (start-stop) 06 (no range, or all)		
10	2	Binary Output—Output status with flags	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
12	1	Binary Command—CROB	3 (select) 4 (operate) 5 (direct op) 6 (dir. op, no ack)	17, 28 (index)	129 (response)	echo of request
20	0	Counter—Any Variation	1 (read) 7 (freeze) 8 (freeze noack) 9 (freeze clear) 10 (frz. cl. noack)	00, 01 (start-stop) 06 (no range, or all)		

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
20	1	Counter—32-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
20	2	Counter—16-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
20	5	Counter—32-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
20	6	Counter—16-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
21	0	Frozen Counter—Any Variation	1 (read) 22 (assign class)	00, 01 (start-stop) 06 (no range, or all)		
21	1	Frozen Counter—32-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
21	2	Frozen Counter—16-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
21	9	Frozen Counter—32-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
21	10	Frozen Counter—16-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
22	0	Counter Event—Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
22	1	Counter Event—32-bit with flag	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
22	2	Counter Event—16-bit with flag	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
23	0	Frozen Counter Event—Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
23	1	Frozen Counter Event—32-bit with flag	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
23	2	Frozen Counter Event—16-bit with flag	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
30	0	Analog Input—Any Variation	1 (read) 22 (assign class)	00, 01 (start-stop) 06 (no range, or all)		
30	1	Analog Input—32-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
30	2	Analog Input—16-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
30	3	Analog Input—32-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
30	4	Analog Input—16-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
32	0	Analog Input Event—Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
32	1	Analog Input Event—32-bit without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
32	2	Analog Input Event—16-bit without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
40	0	Analog Output Status—Any Variation	1 (read)	00, 01 (start-stop) 06 (no range, or all)		
40	1	Analog Output Status—32-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
40	2	Analog Output Status—16-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
41	1	Analog Output—32-bit	3 (select) 4 (operate) 5 (direct op) 6 (dir. op, no ack)	17, 28 (index)	129 (response)	echo of request
41	2	Analog Output—16-bit	3 (select) 4 (operate) 5 (direct op) 6 (dir. op, no ack)	17, 28 (index)	129 (response)	echo of request
50	1	Time and Date—Absolute time	1 (read)	07 (limited qty = 1)	129 (response)	07 (limited qty) (qty = 1)
			2 (write)	07 (limited qty = 1)		
51	1	Time and Date CTO—Absolute time, synchronized			129 (response) 130 (unsol. resp)	07 (limited qty) (qty = 1)
51	2	Time and Date CTO—Absolute time, unsynchronized			129 (response) 130 (unsol. resp)	07 (limited qty) (qty = 1)
52	1	Time Delay—Coarse			129 (response)	07 (limited qty) (qty = 1)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
52	2	Time Delay—Fine			129 (response)	07 (limited qty) (qty = 1)
60	1	Class Objects—Class 0 data	1 (read)	06 (no range,or all)		
			22 (assign class)	06 (no range,or all)		
60	2	Class Objects—Class 1 data	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
			20 (enbl. unsol.) 21 (dab. unsol.) 22 (assign class)	06 (no range,or all)		
60	3	Class Objects—Class 2 data	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
			20 (enbl. unsol.) 21 (dab. unsol.) 22 (assign class)	06 (no range,or all)		
60	4	Class Objects—Class 3 data	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
			20 (enbl. unsol.) 21 (dab. unsol.) 22 (assign class)	06 (no range,or all)		
80	1	Internal Indications—Packed format	1 (read)	00, 01 (start-stop)	129 (response)	00, 01 (start-stop)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
			2 (write)	00 (start-stop) index=7		
No Object (function code only)			13 (cold restart)			
No Object (function code only)			23 (delay meas.)			

NOTE—The following objects were previously required for DNP3 Subset Level 3 conformance and are no longer required as of 2007:

— Binary Command—Pattern control block (g12v2)

— Binary Command—Pattern mask (g12v3)

14.6 Level 4 DNP3 implementation (DNP3-L4)

This subclause describes a subset of the protocol, incorporating all of Level 3 and adding data types and function codes that were found to be useful or were created after definition of the first three original subset levels.

14.6.1 Intended use

This level of implementation of DNP3 is for communicating between a master and a medium-size outstation (e.g., RTU or Data Concentrator). Level 4 may be used where additional capabilities are required.

14.6.2 General description

A Level 4 outstation implementation is the same as a Level 3 implementation with the following additions:

— Self-Address Reservation—If a device receives a message with a destination address of 0xFFFC, and the “self-address” feature is supported and enabled, it shall respond normally with its own source address instead of 0xFFFC.

— Double-bit Input Objects.

— Variations with time for Frozen Counters, Frozen Counter Events, and Analog Input Events.

— Floating-point variations for both Analog Inputs and Analog Outputs.

— Analog Input Reporting Deadband.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— Event Objects for Binary and Analog Outputs.

14.6.3 Implementation table

Table 135 describes the objects, function codes, and qualifiers used in a Level 4 DNP3 implementation.

Table 135 —Level 4 implementation (DNP3-L4)

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
0	211	Device Attributes—Identifier of support for user-specific attributes	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	212	Device Attributes—Number of master-defined data set prototypes	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	213	Device Attributes—Number of outstation-defined data set prototypes	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	214	Device Attributes—Number of master-defined data sets	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	215	Device Attributes—Number of outstation-defined data sets	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	216	Device Attributes—Max number of binary outputs per request	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	217	Device Attributes—Local timing accuracy	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	218	Device Attributes—Duration of timing accuracy	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	219	Device Attributes—Support for analog output events	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
0	220	Device Attributes—Max analog output index	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	221	Device Attributes—Number of analog outputs	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	222	Device Attributes—Support for binary output events	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	223	Device Attributes—Max binary output index	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	224	Device Attributes—Number of binary outputs	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	225	Device Attributes—Support for frozen counter events	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	226	Device Attributes—Support for frozen counters	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	227	Device Attributes—Support for counter events	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	228	Device Attributes—Max counter index	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	229	Device Attributes—Number of counter points	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	230	Device Attributes—Support for frozen analog inputs	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	231	Device Attributes—Support for analog input events	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
0	232	Device Attributes—Maximum analog input index	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	233	Device Attributes—Number of analog input points	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	234	Device Attributes—Support for double-bit binary input events	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	235	Device Attributes—Maximum double-bit binary input index	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	236	Device Attributes—Number of double-bit binary input points	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	237	Device Attributes—Support for binary input events	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	238	Device Attributes—Max binary input index	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	239	Device Attributes—Number of binary input points	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	240	Device Attributes—Max transmit fragment size	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	241	Device Attributes—Max receive fragment size	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	242	Device Attributes—Device manufacturer's software version	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	243	Device Attributes—Device manufacturer's hardware version	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
0	245	Device Attributes—User-assigned location name	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
			2 (write)	00 (start-stop)		
0	246	Device Attributes—User assigned ID code/number	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
			2 (write)	00 (start-stop)		
0	247	Device Attributes—User-assigned device name	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
			2 (write)	00 (start-stop)		
0	248	Device Attributes—Device serial number	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	249	Device Attributes—DNP3 subset and conformance	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	250	Device Attributes—Device manufacturer's product name and model	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	252	Device Attributes—Device manufacturer's name	1 (read)	00 (start-stop)	129 (response)	00 (start-stop) 17 (index)
0	254	Device Attributes—Non-specific all attributes request	1 (read)	00 (start-stop) 06 (no range, or all)		
0	255	Device Attributes—List of attribute variations	1 (read)	00 (start-stop) 06 (no range, or all)	129 (response)	00 (start-stop) 5B (Free Format)
1	0	Binary Input—Any Variation	1 (read) 22 (assign class)	00, 01 (start-stop) 06 (no range, or all)		

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
1	1	Binary Input—Packed format	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
1	2	Binary Input—With flags	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
2	0	Binary Input Event—Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
2	1	Binary Input Event—Without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
2	2	Binary Input Event—With absolute time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
2	3	Binary Input Event—With relative time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
3	0	Double-bit Binary Input—Any Variation	1 (read) 22 (assign class)	00, 01 (start-stop) 06 (no range, or all)		
3	1	Double-bit Binary Input—Packed format	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
3	2	Double-bit Binary Input—With flags	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
4	0	Double-bit Binary Input Event—Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
4	1	Double-bit Binary Input Event—Without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
4	2	Double-bit Binary Input Event—With absolute time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
4	3	Double-bit Binary Input Event—With relative time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
10	0	Binary Output—Any Variation	1 (read)	00, 01 (start-stop) 06 (no range, or all)		
10	2	Binary Output—Output status with flags	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
11	0	Binary Output Event—Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
11	1	Binary Output Event—Status without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
11	2	Binary Output Event—Status with time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
12	1	Binary Command—CROB	3 (select) 4 (operate) 5 (direct op) 6 (dir. op, no ack)	17, 28 (index)	129 (response)	echo of request
13	0	Binary Output Command Event—Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
13	1	Binary Output Command Event—Command status without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
13	2	Binary Output Command Event—Command status with time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
20	0	Counter—Any Variation	1 (read) 7 (freeze) 8 (freeze noack) 9 (freeze clear) 10 (frz. cl. noack)	00, 01 (start-stop) 06 (no range, or all)		
20	1	Counter—32-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
20	2	Counter—16-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
20	5	Counter—32-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
20	6	Counter—16-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
21	0	Frozen Counter—Any Variation	1 (read) 22 (assign class)	00, 01 (start-stop) 06 (no range, or all)		
21	1	Frozen Counter—32-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
21	2	Frozen Counter—16-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
21	5	Frozen Counter—32-bit with flag and time	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
21	6	Frozen Counter—16-bit with flag and time	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
21	9	Frozen Counter—32-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
21	10	Frozen Counter—16-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
22	0	Counter Event—Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
22	1	Counter Event—32-bit with flag	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
22	2	Counter Event—16-bit with flag	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
23	0	Frozen Counter Event—Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
23	1	Frozen Counter Event—32-bit with flag	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
23	2	Frozen Counter Event—16-bit with flag	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
23	5	Frozen Counter Event—32-bit with flag and time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
23	6	Frozen Counter Event—16-bit with flag and time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
30	0	Analog Input—Any Variation	1 (read) 22 (assign class)	00, 01 (start-stop) 06 (no range, or all)		
30	1	Analog Input—32-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
30	2	Analog Input—16-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
30	3	Analog Input—32-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
30	4	Analog Input—16-bit without flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
30	5	Analog Input—Single-prec flt-pt with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
32	0	Analog Input Event—Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
32	1	Analog Input Event—32-bit without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
32	2	Analog Input Event—16-bit without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
32	3	Analog Input Event—32-bit with time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
32	4	Analog Input Event—16-bit with time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
32	5	Analog Input Event—Single-prec flt-pt without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
32	7	Analog Input Event—Single-prec flt-pt with time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
34	0	Analog Input Deadband—Any Variation	1 (read)	00, 01 (start-stop) 06 (no range, or all)		
34	1	Analog Input Deadband—16-bit	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
			2 (write)	00, 01 (start-stop)		
34	2	Analog Input Deadband—32-bit	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
			2 (write)	00, 01 (start-stop)		
34	3	Analog Input Deadband—Single-prec flt-pt	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
			2 (write)	00, 01 (start-stop)		
40	0	Analog Output Status—Any Variation	1 (read)	00, 01 (start-stop) 06 (no range, or all)		
40	1	Analog Output Status—32-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
40	2	Analog Output Status—16-bit with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
40	3	Analog Output Status—Single-prec flt-pt with flag	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
41	1	Analog Output—32-bit	3 (select) 4 (operate) 5 (direct op) 6 (dir. op, no ack)	17, 28 (index)	129 (response)	echo of request
41	2	Analog Output—16-bit	3 (select) 4 (operate) 5 (direct op) 6 (dir. op, no ack)	17, 28 (index)	129 (response)	echo of request
41	3	Analog Output—Single-prec flt-pt	3 (select) 4 (operate) 5 (direct op) 6 (dir. op, no ack)	17, 28 (index)	129 (response)	echo of request
42	0	Analog Output Event—Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
42	1	Analog Output Event—32-bit without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
42	2	Analog Output Event—16-bit without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
42	3	Analog Output Event—32-bit with time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
42	4	Analog Output Event—16-bit with time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
42	5	Analog Output Event—Single-prec flt-pt without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
42	7	Analog Output Event—Single-prec flt-pt with time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
43	0	Analog Output Command Event– Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
43	1	Analog Output Command Event—32-bit without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
43	2	Analog Output Command Event—16-bit without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
43	3	Analog Output Command Event—32-bit with time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
43	4	Analog Output Command Event—16-bit with time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
43	5	Analog Output Command Event—Single-prec flt-pt without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
43	7	Analog Output Command Event—Single-prec flt-pt with time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
50	1	Time and Date—Absolute time	1 (read)	07 (limited qty = 1)	129 (response)	07 (limited qty) (qty = 1)
			2 (write)	07 (limited qty = 1)		
50	3	Time and Date—Absolute time at last recorded time	2 (write)	07 (limited qty = 1)		
51	1	Time and Date CTO—Absolute time, synchronized			129 (response) 130 (unsol. resp)	07 (limited qty) (qty = 1)
51	2	Time and Date CTO—Absolute time, unsynchronized			129 (response) 130 (unsol. resp)	07 (limited qty) (qty = 1)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
52	1	Time Delay—Coarse			129 (response)	07 (limited qty) (qty = 1)
52	2	Time Delay—Fine			129 (response)	07 (limited qty) (qty = 1)
60	1	Class Objects—Class 0 data	1 (read)	06 (no range,or all)		
			22 (assign class)	06 (no range,or all)		
60	2	Class Objects—Class 1 data	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
			20 (enbl. unsol.) 21 (dab. unsol.) 22 (assign class)	06 (no range,or all)		
60	3	Class Objects—Class 2 data	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
			20 (enbl. unsol.) 21 (dab. unsol.) 22 (assign class)	06 (no range,or all)		
60	4	Class Objects—Class 3 data	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
			20 (enbl. unsol.) 21 (dab. unsol.) 22 (assign class)	06 (no range,or all)		

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
80	1	Internal Indications—Packed format	1 (read)	00, 01 (start-stop)	129 (response)	00, 01 (start-stop)
			2 (write)	00 (start-stop) index=7		
No Object (function code only)			13 (cold restart)			
No Object (function code only)			23 (delay meas.)			

14.7 Conformance

This subclause specifies the functions a master or an outstation shall support in order to conform to a defined implementation level.

The subset definitions deal primarily with the Application Layer of the DNP3 protocol. It is nevertheless a requirement that in order for a device to conform to an implementation level, it shall have implemented on it services of the Data Link Layer and Transport Functions (as defined in Clause 8 and Clause 9) sufficient to support the implementation level.

14.7.1 Outstation devices

In order for an outstation to implement a particular Level "X" of DNP3, the device shall conform to the following:

— The outstation shall be able to parse all master requests defined for Level "X."

— The outstation shall be configurable to not transmit anything other than Level "X" responses to Level "X" requests.

— The vendor shall describe the device's DNP3 implementation in a Device Profile Document and make this document available to users of the device.

The Device Profile Document consists of a form, an implementation table similar to those found in this document, and some optional accompanying text. See the Device Profile: Appendix 1 of Volume 8 of the DNP3 Specification [B1] for details of the device profile format.

An outstation may accept and respond to additional *requests* not defined in Level "X" and still conform to that level. It may respond to such requests with data not defined in Level "X."

Any additional functions the outstation provides shall not prevent a master device from communicating with an outstation on the defined level. For instance, a DNP3-L1 outstation may choose to accept WRITE requests of File Identifier Objects. However, it shall not *require* a master to send such a request in order to operate.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

14.7.2 Master devices

In order for a master device to implement a particular level "X" of DNP3, the device shall conform to the following:

— The master device shall be able to parse all outstation responses defined for Level "X."

— The master device shall be configurable to limit the requests it sends to outstations with implementation levels lower than "X." For instance, a Level 3 master shall be configurable so it does not send any requests to either a Level 1 outstation or a Level 2 outstation that it could not parse. This does not prevent the master from sending Level 3 requests to a Level 3 outstation.

— The vendor of the master device shall describe the device's DNP3 implementation in a Device Profile Document and make this document available to users of the device.

The Device Profile Document consists of a form, an implementation table similar to those found in this document, and some optional accompanying text. See the Device Profile: Appendix 1 of Volume 8 of the DNP3 Specification [B1].

14.8 XML representation

The original version of the Device Profile Document only documented how the outstation's capabilities and current settings should be presented in human-readable form. There are many benefits to having this information available in a form that is machine readable as well. These include a more accurate and efficient device setup. This subclause and the associated XML Schema define a mechanism to describe an outstation using the XML.

14.8.1 Background

The XML specification defines a language that can be used to transfer information electronically in a platform- and application-independent fashion. XML defines the syntax for the language but not a valid vocabulary for a specific application. The XML Schema specification provides a mechanism for describing a vocabulary to be used in a specific application.

A detailed description of XML and XML Schemas is beyond the scope of this document. A brief overview, along with a simple example, is presented to assist readers unfamiliar with XML in understanding the remainder of this document, as well as the associated schema.

14.8.1.1 XML

As mentioned, the XML is a specification that defines the syntax for a standardized markup language. A markup language is a language that allows the distribution of data (usually readable text) along with embedded meta-data (data that includes additional and/or qualifying information about the original data).

An XML document is organized as a set of elements. Each element begins with a start tag and ends with an end tag. Start tags have the form '<*element name*>' and end tags have the form '</*element name*>'. Elements can contain other elements (i.e., nested elements) or values. An example of an XML document would be as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<dateOfBirth>
  <day>20</day>
```


Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

```
    <month>March</month>
    <year>1960</year>
  </dateOfBirth>
```

The first line of the document simply identifies the document as an XML document, version 1.0 and using the UTF-8 encoding. This line is fairly standard in all XML documents. The remainder of the document is what we are concerned with. The top-level element in this document is *dateOfBirth*. This element contains three sub-elements named *day*, *month*, and *year*. Each of these elements contains text that provides the value for this element. Elements can be nested as required; they can contain multiple lines of text, etc.

As mentioned, XML provides the syntax for a standard markup language, but it does not define the vocabulary to be used. Hence, in the preceding example, the XML specification defines where the brackets are, what is included in the element's content, etc., but the exact elements that can be used (*dateOfBirth*, *day*, *month*, *year*), and the valid values for these elements, are up to the user. Description of the vocabulary to be used in an XML document is typically handled by an XML Schema as discussed in the following subclauses.

14.8.1.2 XML Schemas

XML Schemas have emerged as the leading standard for defining XML vocabularies. An example of an XML Schema that can be used to describe the XML content in the preceding example is as follows:

```
<xs:simpleType name="dayOfMonthType">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="1"/>
    <xs:maxInclusive value="31"/>
  </xs:restriction>
</xs:simpleType>


<xs:simpleType name="nameOfMonthType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="January"/>
    <xs:enumeration value="February"/>
    <xs:enumeration value="March"/>
    <xs:enumeration value="April"/>
    <xs:enumeration value="May"/>
    <xs:enumeration value="June"/>
    <xs:enumeration value="July"/>
    <xs:enumeration value="August"/>
    <xs:enumeration value="September"/>
    <xs:enumeration value="October"/>
    <xs:enumeration value="November"/>
    <xs:enumeration value="December"/>
  </xs:restriction>
</xs:simpleType>
```


Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

```
<xs:complexType name="dateOfBirthType">
  <xs:sequence>
    <xs:element name="day" type="dayOfMonthType"/>
    <xs:element name="month" type="nameOfMonthType"/>
    <xs:element name="year" type="xs:positiveInteger"/>
  </xs:sequence>
</xs:complexType>


<xs:element name="dateOfBirth" type="dateOfBirthType"/>
```

This schema defines two basic data types, a *dayOfMonthType*, which consists of integer values between 1 and 31 inclusive, and a *nameOfMonthType*, which consists of the names of all the months. It then defines a type called *dateOfBirthType*, which consists of three elements *day*, *month*, and *year*. Finally it defines a single top-level element called *dateOfBirth*.

By using the XML Schema to validate the XML content from the preceding example, a program can use standard library functions to verify syntax and vocabulary for any document. This guarantees that all required elements exist and that all the values are valid. This removes the burden of performing data validation from the target application. XML and XML Schemas provide a very powerful mechanism for exchanging information between two systems.

14.8.1.3 XSLT

XSLT defines a system that can be used to transform XML documents into a different document (which may or may not be XML) using a template. A user-defined XSLT template is used by an XSL engine to convert input XML documents to other documents. Frequently this is used to dynamically generate HTML documents from XML data for display on the Web. The DNP User's Group provides an XSLT template that can be used to translate a DNP3 Device Profile XML instance file to an HTML file with contents formatted similar to the Device Profile Document shown in the Device Profile: Appendix 1 of Volume 8 of the DNP3 Specification [B1].

When using the XSLT template, there are two parameters that can be set to change the way that the XML document is presented:

a) <notes> can be set to yes or no to manage the display of any notes that may be included in the XML file. This parameter defaults to yes.

b) <IEC61850Map> is set to manage the presentation of section 2 of the Device Profile. It can be set to none if section 2 is to be omitted from the presentation. It can be set to table to present section 2 in tabular form, or it can be set to tree to present section 2 as a tree structure. This parameter defaults to tree.

14.8.1.4 XML Schema specifications

The XML specification can be found at:

— http://www.w3.org/TR/2004/REC-xml-20040204/

Details on XML Schemas can be found at:

— http://www.w3.org/TR/xmlschema-0

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— http://www.w3.org/TR/xmlschema-1

— http://www.w3.org/TR/xmlschema-2

The XSLT specification can be found at:

— http://www.w3.org/TR/xslt

14.8.2 Use cases

The DNP3 Device Profile Schema distributed with this document uses XML and XML Schemas as described to define a vocabulary that can be used to describe a DNP3 outstation utilizing an approach that parallels the DNP3 Device Profile Document shown in The Device Profile: Appendix 1 of Volume 8 of the DNP3 Specification [B1]. There are many ways in which a document of this type could be used. This subclause shall try to describe a few common ones.

14.8.2.1 Utility compares implementations

A vendor can distribute an XML document that describes the capabilities of an outstation as well as its default configuration. This document could be published on the Web so potential customers could review it and determine whether the device would meet their needs. In addition, a copy of this file could be shipped with each device and used by the customer to configure a master and/or configuration tool to which the device shall be connected.

14.8.2.2 Utility publishes proposed device requirements

A utility that is planning to install new devices could generate an XML document that describes the desired configuration for the new device. This XML document could then be compared against an XML file that describes a prospective device's capabilities to determine whether that device can be configured to meet the utility's needs.

A simple application could be developed that takes the XML document distributed by a vendor that describes a device's capabilities, and another XML document from a utility that describes the utilities requirements, and verifies whether the device can satisfy the requirements.

14.8.2.3 Outstation publishes current configuration

An outstation tool (within the outstation or a separate program running on another computer) can publish its current configuration at any time by generating a new XML configuration file. This file could be distributed using the standard DNP3 file transfer mechanism. Masters can read this file and use it to set up their own configuration and/or database to match the remote device.

14.8.2.4 Master updates outstation configuration

A master might read the XML document provided by a vendor, or read it directly from the device, and use this to present a user interface that shows the current configuration and supported options for a particular device. The user might then modify the configuration and send the new configuration back to the device as a modified XML file.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

14.8.3 DNP3 XML Schema overview

The DNP3 XML Schema mirrors the DNP3 Device Profile Document in terms of which DNP3 parameters are specified, what options are available, etc. Hence, it is not necessary to describe each of the elements of the DNP3 XML Schema in detail here. Simply refer to the description for the corresponding Device Profile item. This subclause shall however outline a few conventions used when implementing the schema, and describe how they should be used when creating an XML instance document for a particular device.

14.8.3.1 Checking for required parameters

Using the DNP3 Device Profile Schema to describe actual vendor devices has shown that it is not possible to develop a schema that mandates the inclusion of all of the XML elements. Experience has shown that:

— Many devices do not support all of the functionality shown in the Device Profile. For example, simple devices may not support network (TCP / UDP) communications, and hence, Section 1.3 of the Device Profile is not relevant. Similarly, older devices may not support secure authentication, and thus Section 1.12 of the Device Profile is not relevant. To cater to these cases, the inclusion of many elements in the schema had to be made optional by setting their attribute minOccurs to zero.

— If the Device Profile is created by the vendor at the time of device manufacture then many settings in the Device Profile are undefined—their definition being part of the configuration process performed by the user at the time of commissioning the device. This again means that many of the schema elements had to be made optional.

The DNP3 Device Profile Schema is therefore defined with virtually all elements optional, the completeness of the Device Profile XML files not being validated by the schema.

14.8.3.2 ReferenceDevice and AuxillaryInfo

The top-level element of the DNP3 Device Profile Schema includes three sub-elements (Figure 90).

Figure 90—Top level of DNP3 Device Profile Schema

The first element, called `documentHeader`, is used to describe the document itself, its title, author, revision history, etc. (Figure 91).

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Figure 91—Example of the Schema's first element

The second element is `referenceDevice`. This element is used to describe the primary view of the device.

Figure 92—Example of Schema's referenceDevice

The third element, called `auxiliaryInfo`, is used to define alternative views of the reference device and is also of type dnp3DeviceOptionalType. The `auxiliaryInfo` element should only contain the elements that are different from the values defined in the `referenceDevice` element. This might be used, for example, to describe a device with multiple communication ports. Multiple `auxiliaryInfo` elements can be included to describe multiple ports.

The elements of type dnp3DeviceOptionalType have an optional attribute description that can be used to describe why this element is being included in the Device Profile. This should be used with each auxiliaryInfo element to describe the device features being defined with the element.

In order to speed up DNP3 file transfer over slow communication lines, unique filenames may be assigned to read predefined portions of the entire file or only configuration parameters that have changed. In a similar manner, the master may write a small file back to the outstation containing only a few parameters to be updated. There are custom user data sections throughout the DNP3 XML file, so application-specific data can also be stored in the same file. If DNP3 XML is supported through DNP3 File I/O, the standard file name dnpDP.xml shall be supported to read a Complete Device Profile Document.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

14.8.3.3 Empty element vs enumeration

A number of the parameters described in the DNP3 Device Profile Schema allow the user to pick from several options. Two mechanisms exist in the XML Schema specification for selecting between a finite set of choices, an enumeration and a choice group. The main difference is that enumerations support selecting between a set of simple types (i.e., a set of strings or a set of numbers) and a choice group supports the selection of one of a group of complex types (i.e., an element that can contain additional elements and/or data). Using a choice group with a set of empty elements is equivalent in functionality to an enumeration with the same options.

An example of an enumeration and its representation in an XML document is as follows:

```
<xs:simpleType name="yesNoType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="yes"/>
    <xs:enumeration value="no"/>
  </xs:restriction>
</xs:simpleType>

<yesNoInstance>yes</yesNoInstance>
```

The same example using a choice group is as follows:

```
<xs:complexType name="yesNoType">
  <xs:choice>
    <xs:element name="yes"/>
    <xs:element name="no"/>
  </xs:choice>
</xs:complexType>

<yesNoInstance><yes/></yesNoInstance>
```

While an enumeration looks a bit cleaner in the instance document, using a choice group allows the user to select between a wider variety of options, including options that contain data. An example is as follows:

```
<xs:complexType name="yesNoType">
  <xs:choice>
    <xs:element name="yes"/>
    <xs:element name="no"/>
    <xs:element name="maybe">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="explanation" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:choice>
</xs:complexType>
```


Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

```
<yesNoInstance>
  <maybe>
    <explanation>This is an explanation</explanation>
  </maybe>
</yesNoInstance>
```

In order to provide the option of selecting items that contain additional information the DNP3 Device Profile Schema routinely uses choice groups, even in some cases where an enumeration would suffice.

14.8.3.4 Per group versus per point parameters

A number of parameters are defined in the Device Profile Object Group tables with the option of "Per Point" along with directions instructing the author to add a column to the point table for this object group if the "Per Point" option is chosen. The DNP3 Device Profile Schema includes optional elements in both the group table and the corresponding point list for these parameters. If the parameter is constant for all the points in a group, the corresponding XML document should only include this parameter in the group table for that data type. If the parameter varies by point, the corresponding XML document should only include this parameter in the point list for that data type. Although it is not recommended, if the element for this parameter does exist at both the group and point levels, the point specification shall override the group specification.

14.8.3.5 Representation of real time DNP3 data

Each data point defined in the DNP3 Device Profile Schema includes a 'dnpData' element that contains sub-elements that represent the real-time data reported via the DNP3 protocol for that data point. The dnpData element is optional and shall typically not be included in DNP3 Device Profile XML instance documents, but its definition is useful for a number of reasons. First, including these elements in the DNP3 Device Profile Schema allows real-time data to be referenced using XPath exactly the same way the rest of the information in the device profile is referenced. Applications can parse these XPath statements and replace them with the real-time data as necessary at runtime. Secondly, including these elements in the DNP3 Device Profile Schema shall also allow the schema to be used to distribute real-time DNP3 data values using XML in situations where this is desirable.

14.8.3.6 User extensions

It is frequently desirable to insert device- or vender-specific information directly into the DNP3 Device Profile instance document. This could be used, for example, to include the entire device configuration in a single XML file that could be accessed via the DNP3 protocol. The following schema element defines a data type used to support the incorporation of user-specific data in a DNP3 Device Profile XML instance document.

```
<!-- Create a type to hold custom user data -->
<xs:complexType name="UserDataType">
  <xs:sequence>
    <xs:any minOccurs="0" maxOccurs="unbounded"
      namespace="##other" processContents="lax"/>
  </xs:sequence>
</xs:complexType>
```


Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

This data type allows the end user to include almost any valid XML content at points in the XML instance document where an element of this type is allowed (see comments on namespaces and validation below). The resulting document shall still validate against the original DNP3 Device Profile XML Schema. An optional userData element of this type is included in all main data structures of the DNP3 Device Profile Schema to allow user specific data to be incorporated at various places within an instance document. An example of the use of this element is as follows:

```
<!-- DNP3 Device Configuration -->
<deviceConfig>
  <vendorName>XYZ Company</vendorName>
  …
  <userData>
    <XYZ xmlns="http://www.example.com/XYZ">
      This is user custom data...
    </XYZ>
  </userData>
</deviceConfig>
```

Note that setting the namespace attribute of the UserDataType element to ##other requires that the custom XML be in a namespace other than the namespace of the containing element. Since the DNP3 Device Profile Schema specifies a target namespace of "http://www.dnp3.org/DNP3DeviceProfileVersion2-xx" (where xx is the minor release version), all custom elements shall specify a namespace, and it shall not be "http://www.dnp3.org/DNP3DeviceProfileVersion2-xx." The preceding example specifies a namespace of http://www.example.com/XYZ for the custom element.

In addition, setting the "processContents" attribute of the "UserDataType" to "lax" instructs the XML parser to validate the custom data against a schema if a schema whose target namespace matches the elements namespace can be found. If no schema for the specified namespace can be found, the custom data is checked for well-formed XML but no schema validation is performed and no errors are thrown. In order to support validation of the custom elements, the XML parser shall support the use of custom schemas (i.e., there shall be some way for the end user to specify custom schemas in addition to the DNP3 Device Profile Schema to be used during validation). In the preceding example, if a schema for the http://www.example.com/XYZ namespace exists and was added to the list of schemas the parser should validate against, the XYZ element would be validated against the XYZ element defined in the schema. If the XYZ element did not exist in the schema, or the schema was not found, no error would be logged.

14.8.4 Mapping DNP3 data points to IEC 61850 Object Models

The IEC 61850 specification defines a rich set of object models that can be used to describe a wide variety of devices used in the utility industry. The option to describe a device in terms of these models, in addition to the traditional DNP3 point map, is desirable.

The IEC 61850 Object Models are described using a hierarchical structure where the leaves of the tree structure are data attributes. Each data attribute has a type such as integer, float, enumeration, string, etc., as well as other attributes that define the data attribute. In order to support a flexible mapping of the DNP3 parameters to an IEC 61850 Object Model, it shall be possible to map any element, or combination of elements, in the DNP3 configuration file to an IEC 61850 data attribute.

The XPath specification defines a flexible and powerful language that can uniquely reference any element and attribute in an XML document. The XPath language also includes a fairly rich set of functions for processing and/or combining multiple elements. The DNP3 to IEC 61850 mapping shall use XPath to specify the DNP3 elements to map to each IEC 61850 data attribute.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

The IEC 61850 specification defines an object identifier that is a string that can be used to uniquely identify any of the data attributes in the IEC 61850 data models. The DNP3 to IEC 61850 mapping shall use this object identifier string to uniquely identify the data attribute being mapped. In addition to the IEC 61850 path, the 61850 data type, functional constraint, common data class, trigger options, and enumeration values can be included in the mapping for each data attribute. These addition parameters provide addition information required to correctly describe the device in terms of an accurate IEC 61850 model.

The following element is used to map one or more DNP3 elements in the DNP3 Device Profile to an IEC 61850 data attribute. An instance of this element is included for each IEC 61850 data attribute.

```
<xs:complexType name="iec61850MappingType">
  <xs:sequence>
    <xs:element name="note" type="xs:string" minOccurs="0"/>
    <xs:element name="dnp3XPath" type="xs:string"/>
    <xs:element name="iec61850Path" type="xs:string"/>
    <xs:element name="functionalConstraint" type="fcEnum"/>
    <xs:element name="dataType" type="basicTypeEnum"/>
    <xs:element name="commonDataClass" type="cdcEnum"/>
    <xs:element name="triggerOptions" type="trgOptEnum" minOccurs="0"/>
    <xs:element name="enumTypeId" type="xs:normalizedString" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

14.9 Instructions for creating a Device Profile Document

The Device Profile: Appendix 1 of Volume 8 of the DNP3 Specification [B1] shows the format of the DNP3 Device Profile Document and DNP3 XML instance file (after applying the XSLT translation to HTML) that can be filled in to describe a DNP3 device, generate a DNP3 functional requirements specification, or many other use cases such as those outlined in 14.8.2.

It is highly recommended that either a word processing document or a DNP3 XML instance file be created, but not both, so that there is not a question of which one is correct. The XML approach is preferred (and takes precedence if both exist) because a human-readable file can easily be generated from the machine-readable file using XSLT.

Note the following special considerations when creating a Device Profile Document:

— An implementation table shall be added to Section 4 of the Device Profile to describe which object groups and variations, function codes, and qualifiers for both requests and responses are supported. Copy the implementation table for the highest subset level supported from the appropriate interoperability Level Implementation section and place it in Section 4 of the Device Profile.

— Additional highlighted rows may be added to indicate functionality supported beyond that subset level. Rows may not be deleted, but instead the font should be changed to ~~strikethrough~~ for any Object Groups not provided by an outstation or not processed by a master (note these Object Groups shall still be parsed).

— If the Device Profile Document is being created for a specific Master Device, the text in the implementation table column headings “Outstation shall parse” and “Outstation may issue” should be deleted, and the remaining text should be revised to state:

REQUEST	RESPONSE
Master can issue	Master parses

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— If the Device Profile Document is being created for a specific Outstation Device, the text in the implementation table column headings "Master may issue" and "Master shall parse" should be deleted, and the remaining text should be revised to state:

REQUEST	RESPONSE
Outstation parses	Outstation can issue

— For more information on the differences between subset levels and other Object Groups that are not part of a subset level, see the Parsing Code chart in Annex A.

— Complete the Revision History Table located at the beginning of the new Device Profile Document.

Special considerations when generating using XML:

— The best method for generating a DNP3 XML instance file is to program the device configuration tool to create this file based on the current device settings.

— The DNP User's Group Web site[39] provides a list of commercially available DNP3 XML tools specifically designed to generate, modify, view, print, and translate to other file formats such as HTML.

— The example DNP3 XML instance files available from the DNP User's Group Web site can be used as a template and modified with any standard text editor or a generic XML editor to create a DNP3 XML instance file for your device.

— The DNP User's Group provides an XML package distributed as a zip file containing the following:

a) A schema file (.xsd) that all DNP3 XML instance files shall validate against

b) An XSLT file that can be used to translate a machine-readable DNP3 XML instance file into an HTML file that can be viewed with a web browser

c) A template XSLT file that can be customised by a vendor to translate those parts of the Device Profile to show userData entries

d) The dnp logo file, used by the XSLT

NOTE—The first three files should be distributed with the DNP3 XML instance file for a device. A DNP3 XML instance file may not be compatible with different release versions of the DNP3 XML Schema and XSLT files.

Special considerations when using the word processing template:

— The general technique for producing a word processing-based Device Profile Document is to fill in the following document with the correct information and to save it to a new file that shall become the Device Profile Document for your device.

— Bookmarks for the Vendor Name, Device Name, and Revision date on the cover page shall automatically update items in the tables of the appendix along with the footer when the cover page is edited.

[39] http://www.dnp.org.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Annex A

(normative)

DNP3 object library

A.1 Data object library—Part two

A.1.1 Insert sheet index

Object		Description	Version
Group	Var		
0	211	Device Attributes—Identifier of support for user-specific attributes	1.00
0	212	Device Attributes—Number of master-defined data set prototypes	1.00
0	213	Device Attributes—Number of outstation-defined data set prototypes	1.00
0	214	Device Attributes—Number of master-defined data sets	1.01
0	215	Device Attributes—Number of outstation-defined data sets	1.01
0	216	Device Attributes—Max number of binary outputs per request	1.00
0	217	Device Attributes—Local timing accuracy	1.00
0	218	Device Attributes—Duration of timing accuraccy	1.00
0	219	Device Attributes—Support for analog output events	1.00
0	220	Device Attributes—Max analog output index	1.00
0	221	Device Attributes—Number of analog outputs	1.00
0	222	Device Attributes—Support for binary output events	1.00
0	223	Device Attributes—Max binary output index	1.00
0	224	Device Attributes—Number of binary outputs	1.00
0	225	Device Attributes—Support for frozen counter events	1.00
0	226	Device Attributes—Support for frozen counters	1.00
0	227	Device Attributes—Support for counter events	1.00
0	228	Device Attributes—Max counter index	1.00
0	229	Device Attributes—Number of counter points	1.00
0	230	Device Attributes—Support for frozen analog inputs	1.00
0	231	Device Attributes—Support for analog input events	1.00
0	232	Device Attributes—Maximum analog input index	1.00
0	233	Device Attributes—Number of analog input points	1.00
0	234	Device Attributes—Support for double-bit binary input events	1.00
0	235	Device Attributes—Maximum double-bit binary input index	1.00
0	236	Device Attributes—Number of double-bit binary input points	1.00
0	237	Device Attributes—Support for binary input events	1.00
0	238	Device Attributes—Max binary input index	1.00
0	239	Device Attributes—Number of binary input points	1.00
0	240	Device Attributes—Max transmit fragment size	1.00
0	241	Device Attributes—Max receive fragment size	1.00
0	242	Device Attributes—Device manufacturer's software version	1.00
0	243	Device Attributes—Device manufacturer's hardware version	1.00
0	244	Not available—Reserved for future assignment	—
0	245	Device Attributes—User-assigned location name	1.00
0	246	Device Attributes—User-assigned ID code/number	1.00
0	247	Device Attributes—User-assigned device name	1.00
0	248	Device Attributes—Device serial number	1.00
0	249	Device Attributes—DNP3 subset and conformance	1.00
0	250	Device Attributes—Device manufacturer's product name and model	1.00

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Object		Description	Version
Group	Var		
0	251	Not available—Reserved for future assignment	—
0	252	Device Attributes—Device manufacturer's name	1.00
0	253	Not available—Reserved for future assignment	—
0	254	Device Attributes— Non-specific all attributes request	1.01
0	255	Device Attributes—List of attribute variations	2.00
1	1	Binary Input—Packed format	0.02
1	2	Binary Input—With flags	0.02
2	1	Binary Input Event—Without time	0.02
2	2	Binary Input Event—With absolute time	0.02
2	3	Binary Input Event—With relative time	0.02
3	1	Double-bit Binary Input—Packed format	1.00
3	2	Double-bit Binary Input—With flags	1.00
4	1	Double-bit Binary Input Event—Without time	1.00
4	2	Double-bit Binary Input Event—With absolute time	1.00
4	3	Double-bit Binary Input Event—With relative time	1.00
10	1	Binary Output—Packed format	0.02
10	2	Binary Output—Output status with flags	0.02
11	1	Binary Output Event—Status without time	1.01
11	2	Binary Output Event—Status with time	1.00
12	1	Binary Command—Control relay output block (CROB)	0.02
12	2	Binary Command—Pattern control block (PCB)	0.02
12	3	Binary Command—Pattern mask	0.02
13	1	Binary Output Command Event—Command status without time	1.00
13	2	Binary Output Command Event—Command status with time	1.00
20	1	Counter—32-bit with flag	0.02
20	2	Counter—16-bit with flag	0.02
20	3	Counter—32-bit with flag, delta	0.02
20	4	Counter—16-bit with flag, delta	0.02
20	5	Counter—32-bit without flag	0.02
20	6	Counter—16-bit without flag	0.02
20	7	Counter—32-bit without flag, delta	0.02
20	8	Counter—16-bit without flag, delta	0.02
21	1	Frozen Counter—32-bit with flag	0.02
21	2	Frozen Counter—16 bit with flag	0.02
21	3	Frozen Counter—32-bit with flag, delta	0.02
21	4	Frozen Counter—16-bit with flag, delta	0.02
21	5	Frozen Counter—32-bit with flag and time	0.02
21	6	Frozen Counter—16-bit with flag and time	0.02
21	7	Frozen Counter—32-bit with flag and time, delta	0.02
21	8	Frozen Counter—16-bit with flag and time, delta	0.02
21	9	Frozen Counter—32-bit without flag	0.02
21	10	Frozen Counter—16-bit without flag	0.02
21	11	Frozen Counter—32-bit without flag, delta	0.02
21	12	Frozen Counter—16-bit without flag, delta	0.02
22	1	Counter Event—32-bit with flag	0.02
22	2	Counter Event—16-bit with flag	0.02
22	3	Counter Event—32-bit with flag, delta	0.02
22	4	Counter Event—16-bit with flag, delta	0.02
22	5	Counter Event—32-bit with flag and time	0.02
22	6	Counter Event—16-bit with flag and time	0.02
22	7	Counter Event—32-bit with flag and time, delta	0.02
22	8	Counter Event—16-bit with flag and time, delta	0.02
23	1	Frozen Counter Event—32-bit with flag	0.02
23	2	Frozen Counter Event—16-bit with flag	0.02
23	3	Frozen Counter Event—32-bit with flag, delta	0.02
23	4	Frozen Counter Event—16-bit with flag, delta	0.02

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Object		Description	Version
Group	Var		
23	5	Frozen Counter Event—32-bit with flag and time	0.02
23	6	Frozen Counter Event—16-bit with flag and time	0.02
23	7	Frozen Counter Event—32-bit with flag and time, delta	0.02
23	8	Frozen Counter Event—16-bit with flag and time, delta	0.02
30	1	Analog Input—32-bit with flag	0.02
30	2	Analog Input—16-bit with flag	0.02
30	3	Analog Input—32-bit without flag	0.02
30	4	Analog Input—16-bit without flag	0.02
30	5	Analog Input—Single-prec flt-pt with flag	0.02
30	6	Analog Input—Double-prec flt-pt with flag	0.02
31	1	Frozen Analog Input—32-bit with flag	0.02
31	2	Frozen Analog Input—16-bit with flag	0.02
31	3	Frozen Analog Input—32-bit with time-of-freeze	0.02
31	4	Frozen Analog Input—16-bit with time-of-freeze	0.02
31	5	Frozen Analog Input—32-bit without flag	0.02
31	6	Frozen Analog Input—16-bit without flag	0.02
31	7	Frozen Analog Input—Single-prec flt-pt with flag	0.02
31	8	Frozen Analog Input—Double-prec flt-pt with flag	0.02
32	1	Analog Input Event—32-bit without time	0.02
32	2	Analog Input Event—16-bit without time	0.02
32	3	Analog Input Event—32-bit with time	0.02
32	4	Analog Input Event—16-bit with time	0.02
32	5	Analog Input Event—Single-prec flt-pt without time	0.02
32	6	Analog Input Event—Double-prec flt-pt without time	0.02
32	7	Analog Input Event—Single-prec flt-pt with time	0.02
32	8	Analog Input Event—Double-prec flt-pt with time	0.02
33	1	Frozen Analog Input Event—32-bit without time	0.02
33	2	Frozen Analog Input Event—16-bit without time	0.02
33	3	Frozen Analog Input Event—32-bit with time	0.02
33	4	Frozen Analog Input Event—16-bit with time	0.02
33	5	Frozen Analog Input Event—Single-prec flt-pt without time	0.02
33	6	Frozen Analog Input Event—Double-prec flt-pt without time	0.02
33	7	Frozen Analog Input Event—Single-prec flt-pt with time	0.02
33	8	Frozen Analog Input Event—Double-prec flt-pt with time	0.02
34	1	Analog Input Reporting Deadband—16-bit	0.01
34	2	Analog Input Reporting Deadband—32-bit	0.01
34	3	Analog Input Reporting Deadband—Single-prec flt-pt	0.01
40	1	Analog Output Status—32-bit with flag	0.02
40	2	Analog Output Status—16-bit with flag	0.02
40	3	Analog Output Status—Single-prec flt-pt with flag	0.02
40	4	Analog Output Status—Double-prec flt-pt with flag	0.02
41	1	Analog Output—32-bit	0.02
41	2	Analog Output—16-bit	0.02
41	3	Analog Output—Single-prec flt-pt	0.02
41	4	Analog Output—Double-prec flt-pt	0.02
42	1	Analog Output Event—32-bit without time	1.01
42	2	Analog Output Event—16-bit without time	1.00
42	3	Analog Output Event—32-bit with time	1.00
42	4	Analog Output Event—16-bit with time	1.00
42	5	Analog Output Event—Single-prec flt-pt without time	1.00
42	6	Analog Output Event—Double-prec flt-pt without time	1.00
42	7	Analog Output Event—Single-prec flt-pt with time	1.00
42	8	Analog Output Event—Double-prec flt-pt with time	1.00
43	1	Analog Output Command Event—32-bit without time	1.00
43	2	Analog Output Command Event—16-bit without time	1.00
43	3	Analog Output Command Event—32-bit with time	1.00

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Object		Description	Version
Group	Var		
43	4	Analog Output Command Event—16-bit with time	1.00
43	5	Analog Output Command Event—Single-prec flt-pt without time	1.00
43	6	Analog Output Command Event—Double-prec flt-pt without time	1.00
43	7	Analog Output Command Event—Single-prec flt-pt with time	1.00
43	8	Analog Output Command Event—Double-prec flt-pt with time	1.00
50	1	Time and Date—Absolute time	0.02
50	2	Time and Date—Absolute time and interval	0.02
50	3	Time and Date—Absolute time at last recorded time	1.00
51	1	Time and Date CTO—Absolute time, synchronized	0.02
51	2	Time and Date CTO—Absolute time, unsynchronized	0.02
52	1	Time Delay—Coarse	0.02
52	2	Time Delay—Fine	0.02
60	1	Class Objects—Class 0 data	1.00
60	2	Class Objects—Class 1 data	1.00
60	3	Class Objects—Class 2 data	1.00
60	4	Class Objects—Class 3 data	1.00
70	1	File-Control—File identifier (superceded)	0.02
70	2	File-Control—Authentication	1.00
70	3	File-Control—File command	1.00
70	4	File-Control—File command status	1.00
70	5	File-Control—File transport	1.00
70	6	File-Control—File transport status	1.00
70	7	File-Control—File descriptor	1.00
70	8	File-Control—File specification string	1.00
80	1	Internal Indications—Packed format	0.02
81	1	Device Storage—Buffer fill status	1.00
82	1	Device Profile—Functions and indexs	0.02
83	1	Data Set—Private registration object	0.02
83	2	Data Set—Private registration object descriptor	0.02
85	1	Data Set Prototype—With UUID	1.00
86	1	Data Set Descriptor—Data set contents	1.00
86	2	Data Set Descriptor—Characteristics	1.00
86	3	Data Set Descriptor—Point index attributes	1.01
87	1	Data Set—Present value	1.00
88	1	Data Set—Snapshot	1.00
90	1	Application—Identifier	0.02
91	1	Status of Requested Operation—Activate Configuration	1.00
100	1–3	Floating-point	1.00
101	1	Binary-Coded Decimal Integer—Small	0.02
101	2	Binary-Coded Decimal Integer—Medium	0.02
101	3	Binary-Coded Decimal Integer—Large	0.02
102	1	Unsigned Integer—8-bit	1.00
110	1–255	Octet String	0.01
111	1–255	Octet String Event	0.01
112	1–255	Virtual Terminal Output Block	0.02
113	1–255	Virtual Terminal Event Data	0.02
120	1	Authentication—Challenge	6.00
120	2	Authentication—Reply	6.00
120	3	Authentication—Aggressive Mode Request	6.00
120	4	Authentication—Session Key Status Request	6.00
120	5	Authentication—Session Key Status	6.00
120	6	Authentication—Session Key Change	6.00
120	7	Authentication—Error	7.00
120	9	Authentication—Hashed Message Authentication Code (HMAC)	5.00
NOTE—Objects on shaded lines are obsolete.			

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2 Object format inserts

Each object group and variation is described in an individual insert consisting of one or more pages. The inserts are organized by object group number and then within the group number by variation number.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.1 Device attributes—Identification of support for user-specific attributes

DNP3 Object Library		Group:	0
		Variation:	211
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Identification of support for user-specific attributes		

Description

This attribute is used to identify the support for private, vendor, or user-specific sets of attributes (collectively referenced as user-specific attributes).

This object attribute consists of an ASCII string that lists all the attribute sets supported by the outstation. The list specifies 0, 1, or more attribute sets.

Each element in the list consists of a namespace–index pair, where the namespace identifies the attribute set and the index specifies at which group 0 index the master can read attributes from that attribute set.

Namespaces are text names that uniquely in the world identify one attribute set from another. Namespaces shall be registered with the DNP User's Group, which is achieved by completing a form on their Web site.[40]

The index part of the namespace–index pair specifies which group 0 index shall be used by the master to access attributes associated with the respective attribute set.

Before attempting to read user-specific attributes, the master shall read g0v211 at index 0. Upon receiving the outstation's response, the master parses the list. Using the information obtained after parsing, the master can unambiguously read or write, as appropriate, attributes associated with a specific attribute set.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.) For this attribute number, this shall be 1 (VSTR).

[40] http://www.dnp.org.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT8: Length.

Specifies the number of octets in the identifying string.

VSTRn: Identification of user-specific attribute sets.

Consists of namespace–index number pairs and delimiters as follows:

SET of 1 (Included for the first attribute set in the list)

{

VSTRx: Namespace.

VSTR1: Comma delimiter used to separate the namespace from the index number fields.

VSTRy: The group 0 index number used to access attributes for this attribute set stated as a decimal value in ASCII characters.

}

SET of m (Included for the second and subsequent attribute sets in the list)

{

VSTR1: Semicolon delimiter used to separate one attribute set from another. The first attribute set omits this field and all others include this field.

VSTRx: Namespace.

VSTR1: Comma delimiter used to separate the namespace from the index number fields.

VSTRy: The group 0 index number used to access attributes for this attribute set stated as a decimal value in ASCII characters.

}

NOTE—If no attribute sets are supported, the data type code is set to 254, the length is set to 0, and there are no octets in the identification string.

Examples of the use of this attribute are

An outstation supporting a singe user-specific attribute set

►►► Request Message

C3	01	00	D3	00	00	00
AC	FC	Grp	Var	Qual	Range	

◄◄◄ Response Message (beginning)

C3	81	00	00	00	D3	17	01	00	01	0A
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Index	Type	Length

◄◄◄ Continuation of Response Message

41	42	43	20	43	6F	72	70	2C	31
←	Namespace of the attribute set						→	,	index

In response to a read request for g0v211 with index 0, an outstation returns the information that attributes associated with the namespace “ABC Corp” are available using index number 1.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

An outstation supporting two user-specific attribute sets

►►► Request Message

C3	01	00	D3	00	00	00
AC	FC	Grp	Var	Qual	Range	

◄◄◄ Response Message (beginning)

C3	81	00	00	00	D3	17	01	00	01	14
AC	FC	IIN_1	IIN_2	Grp	Var	Qual	Range	Index	Type	Length

◄◄◄ Continuation of Response Message

41	42	43	20	43	6F	72	70	2C	31	3B
←	Namespace of the 1st attribute set						→	,	Index	;

◄◄◄ Continuation of Response Message

58	59	5A	20	49	6E	63	2C	32
←	Namespace of the 2nd attribute set					→	,	Index

In response to a read request for g0v211 with index 0, an outstation returns the information that attributes associated with the namespace “ABC Corp” are available using index number 1 and those for namespace “XYZ Inc” are available using index number 2.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.2 Device attributes—Number of master-defined data set prototypes

DNP3 Object Library		Group:	0
		Variation:	212
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Number of master-defined data set prototypes		

Description

This attribute provides the number of data set prototypes in the outstation that were defined by the master.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Number of master-defined data set prototypes* field.

UINTn: Number of master-defined data set prototypes.

The number of data set prototypes that exist in the outstation when this object is reported that were defined by the master.

NOTE 1—When an outstation starts, it may have N outstation-defined data set prototypes with indexes 0 to N – 1. This corresponds to the number reported in object group 0, variation 213. Master-defined data set prototypes begin their index numbering with the number following the highest outstation-defined index; N for the preceding example.

NOTE 2—Outstations that accept master-defined data set prototypes, but that store their definition in volatile memory, may start up with zero master-defined data set prototypes. The value reported by this object shall change after the master writes data set prototypes to the outstation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.3 Device attributes—Number of outstation-defined data set prototypes

DNP3 Object Library		Group:	0
		Variation:	213
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Number of outstation-defined data set prototypes		

Description

This attribute provides the number of data set prototypes in the outstation that were defined by the outstation.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Number of outstation-defined data set prototypes* field.

UINTn: Number of outstation-defined data set prototypes.

The number of data set prototypes in the outstation that exist when this object is reported that were defined by the outstation.

NOTE—When an outstation starts, it may have N outstation-defined data set prototypes with indexes 0 to N – 1. This corresponds to the number reported by this object. Master-defined data set prototypes begin their index numbering with the number following the highest outstation-defined index; N for the preceding example.

Version: 1.00	Approval Date: 24 September 2005	DNPObjLibIns_G00_V213	Sheet 1

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.4 Device attributes—Number of master-defined data sets

DNP3 Object Library		Group:	0
		Variation:	214
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Number of master-defined data sets		

Description

This attribute provides the number of data sets in the outstation that were defined by the master.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Number of master-defined data sets* field.

UINTn: Number of master-defined data sets.

The number of data sets in the outstation when this object is reported that were defined by the master.

NOTE 1—When an outstation starts, it may have N outstation-defined data sets with indexes 0 to N – 1. This corresponds to the number reported in object group 0, variation 215. Master-defined data sets begin their index numbering with the number following the highest outstation-defined index; N for the preceding example.

NOTE 2—Outstations that accept master-defined data sets, but that store their definition in volatile memory, may start up with zero master-defined data sets. The value reported by this object should change after the master writes data set descriptors to the outstation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.5 Device attributes—Number of outstation-defined data sets

DNP3 Object Library		Group:	0
		Variation:	215
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Number of outstation-defined data sets		

Description

This attribute provides the number of data sets in the outstation that were defined by the outstation.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position	
									Attribute data type code
									Length
									Number of outstation-defined data sets

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Number of outstation-defined data sets* field.

UINTn: Number of outstation-defined data sets.

The number of data sets in the outstation when this object is reported that were defined by the outstation.

NOTE—When an outstation starts, it may have N outstation-defined data sets with indexes 0 to N – 1. This corresponds to the number reported by this object. Master-defined data sets begin their index numbering with the number following the highest outstation-defined index; N for the preceding example.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.6 Device attributes—Maximum number of binary output objects per request

DNP3 Object Library		Group:	0
		Variation:	216
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Maximum number of binary output objects per request		

Description

This attribute is the maximum number of binary output objects from object group 12 that the master may include in control messages.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Number of binary output objects* field.

UINTn: Maximum number of binary output objects.

The maximum number of binary output objects from object group 12 that the master may include in control messages.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.7 Device attributes—Local timing accuracy

DNP3 Object Library		Group:	0
		Variation:	217
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Local timing accuracy		

Description

This attribute is the rated timing accuracy of an outstation stated in microseconds. This value applies to the maximum error in the detection and time stamping of events when the time for those events is determined by the outstation. It does not apply to events that are retrieved and passed on from other devices when the event time is determined by those other devices.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Timing accuracy microseconds* field.

UINTn: Timing accuracy microseconds.

This value is defined as the maximum difference in time, stated as a positive number of microseconds, between the time that would be given to an event by a hypothetical device having a perfect clock with infinitesimal resolution and no detection delays and the worst-case time that the outstation could give.

NOTE 1—Do not confuse this attribute with timing resolution. Resolution is the smallest interval of time between data sampling or computations or reported times. DNP3 events are reported with a time resolution of one millisecond. An outstation, for example, might be able to detect the existence of an event every 100 microseconds, but that does not necessarily mean that its timing accuracy is 100 microseconds. This hypothetical outstation's timing accuracy could be much worse, say, 20 000 microseconds.

NOTE 2—When reporting this value, the calculations should take into consideration the drift in the device's local clock source, the period between time synchronizations, sampling resolution, and other factors.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

NOTE 3—If an outstation requires time synchronization from a master and asserts the Internal Indications bit IIN1.4 [NEED_TIME], the calculation should assume that the time reported by the master is perfect. The timing accuracy then depends on

— The drift in the outstation's local oscillator and the period between assertions of bit IIN1.4.

— How well the outstation can determine the time delay when a delay measurement (function code 23 [DELAY_MEASURE]) is requested from the master.

— Interrupt latency.

— Data sampling periods.

— Filter delays.

— Other factors.

NOTE 4—If an outstation is synchronized from a local source, such as a GPS receiver, the time accuracy reported should include the error in that product and other possible error sources such as those listed in the previous paragraph.

NOTE 5—If an outstation contains multiple timing sources, only one is reportable using this object. It is suggested that vendors choose the source associated with the most critical measurements, those most likely to be used for forensic analysis should a major event occur. As an example, given a protective relay with independent time tagging sources for faults and analog measurements, the source associated with breaker tripping would be chosen.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.8 Device attributes—Duration of time accuracy

DNP3 Object Library		Group:	0
		Variation:	218
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Duration of time accuracy		

Description

This attribute is the number of seconds that the device maintains its rated time accuracy following a time synchronization from the master.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Number of seconds* field.

UINTn: Number of seconds.

The number of seconds that the device maintains its rated time accuracy following a time synchronization from the master.

If time synchronization is not required from the master, this value shall be zero. This includes devices that do not time tag events and devices that receive time from other sources such as a GPS receiver.

If time in the device is dependent on periodic receipt of time from the master, then this value shall be non-zero. This value shall not exceed 0xFFFFFFFF.

NOTE 1—Rated accuracy is not defined by DNP3 as this is a vendor- and application-specific parameter.

NOTE 2—If an outstation contains multiple timing sources, only one is reportable using this object. It is suggested that vendors choose the source associated with the most critical measurements, those most likely to be used for forensic analysis should a major event occur. As an example, given a protective relay with independent time tagging sources for faults and analog measurements, the source associated with breaker tripping would be chosen.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.9 Device attributes—Support for analog output events

DNP3 Object Library		Group:	0
		Variation:	219
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Support for analog output events		

Description

This attribute is Boolean that indicates whether the device supports analog output events.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position	
									Attribute data type code
									Length
									Analog output events

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, INT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Analog output events* field.

INTn: Analog output events.

This is a Boolean having a value of

1 if the device supports analog output events and

0 if the device does not support analog output events.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.10 Device attributes—Maximum analog output index

DNP3 Object Library		Group:	0
		Variation:	220
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Maximum analog output index		

Description

This attribute is maximum analog output point index controllable from the master.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position	
									Attribute data type code
									Length
									Maximum analog output point index

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Maximum analog output point index* field.

UINTn: Maximum analog output point index.

The presently configured maximum analog output point index controllable from the master.

NOTE—If there are gaps in the point indexes, the number of points controllable by the master in variation 221 might not equal one more than the maximum point index.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.11 Device attributes—Number of analog outputs

DNP3 Object Library		Group:	0
		Variation:	221
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Number of analog outputs		

Description

This attribute is number of analog output points controllable from the master.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position	
									Attribute data type code
									Length
									Number of analog output points

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Number of analog output points* field.

UINTn: Number of analog output points.

The presently configured number of analog output points that are controllable from the master.

NOTE—If there are gaps in the point indexes, the number of points controllable by the master might not equal one more than the maximum point index in variation 220.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.12 Device attributes—Support for binary output events

DNP3 Object Library		Group:	0
		Variation:	222
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Support for binary output events		

Description

This attribute is Boolean that indicates whether the device supports binary output events.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position	
									Attribute data type code
									Length
									Binary output events

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, INT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Binary output events* field.

INTn: Binary output events.

This is a Boolean having a value of

1 if the device supports binary output events and

0 if the device does not support binary output events.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.13 Device attributes—Maximum binary output index

DNP3 Object Library		Group:	0
		Variation:	223
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Maximum binary output index		

Description

This attribute is maximum binary output point index controllable from the master.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Maximum binary output point index* field.

UINTn: Maximum binary output point index.

The presently configured maximum binary output point index controllable from the master.

NOTE—If there are gaps in the point indexes, the number of points controllable from the master in variation 224 might not equal one more than the maximum point index.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.14 Device attributes—Number of binary outputs

DNP3 Object Library		Group:	0
		Variation:	224
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Number of binary outputs		

Description

This attribute is number of binary output points controllable from the master.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position	
									Attribute data type code
									Length
									Number of binary output points

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Number of binary output points* field.

UINTn: Number of binary output points.

The presently configured number of binary output points controllable from the master.

NOTE—If there are gaps in the point indexes, the number of points controllable from the master might not equal one more than the maximum point index in variation 223.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.15 Device attributes—Support for frozen counter events

DNP3 Object Library		Group:	0
		Variation:	225
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Support for frozen counter events		

Description

This attribute is Boolean that indicates whether the device supports frozen counter events.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position	
									Attribute data type code
									Length
									Frozen counter events

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, INT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Frozen counter events* field.

INTn: Frozen counter events.

This is a Boolean having a value of

1 if the device supports frozen counter events and

0 if the device does not support frozen counter events.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.16 Device attributes—Support for frozen counters

DNP3 Object Library		Group:	0
		Variation:	226
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Support for frozen counters		

Description

This attribute is Boolean that indicates whether the device supports frozen counters.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position	
									Attribute data type code
									Length
									Frozen counters

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, INT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Frozen counters* field.

INTn: Frozen counters.

This is a Boolean having a value of

1 if the device supports frozen counters and

0 if the device does not support frozen counters.

.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.17 Device attributes—Support for counter events

DNP3 Object Library		Group:	0
		Variation:	227
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Support for counter events		

Description

This attribute is Boolean that indicates whether the device supports counter events.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position	
									Attribute data type code
									Length
									Counter events

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, INT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Counter events* field.

INTn: Counter events.

This is a Boolean having a value of

1 if the device supports counter events and

0 if the device does not support counter events.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.18 Device attributes—Maximum counter index

DNP3 Object Library		Group:	0
		Variation:	228
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Maximum counter index		

Description

This attribute is maximum counter point index reported by the device.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
								Attribute data type code
								Length
								Maximum counter point index

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Maximum counter point index* field.

UINTn: Maximum counter point index.

The presently configured maximum counter point index reported by the device.

NOTE—If there are gaps in the point indexes, the number of points reported in variation 229 might not equal one more than the maximum point index.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.19 Device attributes—Number of counter points

DNP3 Object Library		Group:	0
		Variation:	229
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Number of counter points		

Description

This attribute is number of counter points reported by the device.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Number of counter points* field.

UINTn: Number of counter points.

The presently configured number of counter points reported by the device.

NOTE—If there are gaps in the point indexes, the number of points reported might not equal one more than the maximum point index reported in variation 228.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.20 Device attributes—Support for frozen analog inputs

DNP3 Object Library		Group:	0
		Variation:	230
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Support for frozen analog inputs		

Description

This attribute is Boolean that indicates whether the device supports frozen analog inputs.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position	
									Attribute data type code
									Length
									Frozen analog inputs

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, INT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Frozen analog inputs* field.

INTn: Frozen analog inputs.

This is a Boolean having a value of

1 if the device supports frozen analog inputs and

0 if the device does not support frozen analog inputs.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.21 Device attributes—Support for analog input events

DNP3 Object Library		Group:	0
		Variation:	231
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Support for analog input events		

Description

This attribute is Boolean that indicates whether the device supports analog input events.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
								Attribute data type code
								Length
								Analog input events

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, INT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Analog input events* field.

INTn: Analog input events.

This is a Boolean having a value of

1 if the device supports analog input events and

0 if the device does not support analog input events.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.22 Device attributes—Maximum analog input index

DNP3 Object Library		Group:	0
		Variation:	232
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Maximum analog input index		

Description

This attribute is maximum analog input point index reported by the device.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Maximum analog input point index* field.

UINTn: Maximum analog input point index.

The presently configured maximum analog input point index reported by the device.

NOTE—If there are gaps in the point indexes, the number of points reported in variation 233 might not equal one more than the maximum point index.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.23 Device attributes—Number of analog input points

DNP3 Object Library		Group:	0
		Variation:	233
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Number of analog input points		

Description

This attribute is number of analog input points reported by the device.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in *the Number of analog input points* field.

UINTn: Number of analog input points.

The presently configured number of analog input points reported by the device.

NOTE—If there are gaps in the point indexes, the number of points reported might not equal one more than the maximum point index reported in variation 232.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.24 Device attributes—Support for double-bit binary input events

DNP3 Object Library		Group:	0
		Variation:	234
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Support for double-bit binary input events		

Description

This attribute is Boolean that indicates whether the device supports double-bit binary input events.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position	
									Attribute data type code
									Length
									Double-bit binary input events

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, INT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Double-bit binary events* field.

INTn: Double-bit binary input events.

This is a Boolean having a value of

1 if the device supports double-bit binary events and

0 if the device does not support double-bit input events.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.25 Device attributes—Maximum double-bit binary index

DNP3 Object Library		Group:	0
		Variation:	235
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Maximum double-bit binary index		

Description

This attribute is maximum double-bit input point index reported by the device.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Maximum double-bit binary input point index* field.

UINTn: Maximum double-bit binary point index.

The presently configured maximum double-bit binary input index reported by the device.

NOTE—If there are gaps in the point indexes, the number of points reported in variation 236 might not equal one more than the maximum point index.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.26 Device attributes—Number of double-bit binary input points

DNP3 Object Library		Group:	0
		Variation:	236
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Number of double-bit binary input points		

Description

This attribute is number of double-bit binary input points reported by the device.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Number of double-bit binary input points* field.

UINTn: Number of double-bit input points.

The presently configured number of double-bit input points reported by the device.

NOTE—If there are gaps in the point indexes, the number of points reported might not equal one more than the maximum point index reported in variation 235.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.27 Device attributes—Support for binary input events

DNP3 Object Library		Group:	0
		Variation:	237
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Support for binary input events		

Description

This attribute is Boolean that indicates whether the device supports binary input events.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position	
									Attribute data type code
									Length
									Binary input events

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, INT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Binary input events* field.

INTn: Binary input events.

This is a Boolean having a value of

1 if the device supports binary input events and

0 if the device does not support binary input events.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.28 Device attributes—Maximum binary input index

DNP3 Object Library		Group:	0
		Variation:	238
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Maximum binary input index		

Description

This attribute is maximum binary input point index reported by the device.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Maximum binary input point index* field.

UINTn: Maximum binary input point index.

The presently configured maximum binary input point index reported by the device.

NOTE—If there are gaps in the point indexes, the number of points reported in variation 239 might not equal one more than the maximum point index.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.29 Device attributes—Number of binary input points

DNP3 Object Library		Group:	0
		Variation:	239
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Number of binary input points		

Description

This attribute is number of binary input points reported by the device.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Number of binary input points* field.

UINTn: Number of binary input points.

The presently configured number of binary input points reported by the device.

NOTE—If there are gaps in the point indexes, the number of points reported might not equal one more than the maximum point index reported in variation 238.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.30 Device attributes—Maximum transmit fragment size

DNP3 Object Library		Group:	0
		Variation:	240
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Maximum transmit fragment size		

Description

This attribute is maximum number of octets the device shall transmit in an Application Layer fragment.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Maximum transmit fragment size* field.

UINTn: Maximum transmit fragment size.

The maximum number of octets the device shall transmit in an Application Layer fragment.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.31 Device attributes—Maximum receive fragment size

DNP3 Object Library		Group:	0
		Variation:	241
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Maximum receive fragment size		

Description

This attribute is maximum number of octets the device shall accept in a received Application Layer fragment.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, UINT. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the *Maximum receive fragment size* field.

UINTn: Maximum receive fragment size.

The maximum number of octets the device shall accept in a received Application Layer fragment.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.32 Device attributes—Device manufacturer's software version

DNP3 Object Library		Group:	0
		Variation:	242
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Device manufacturer's software version		

Description

This attribute is the version code of the manufacturer's device software.

The contents of this attribute is a free form string that is formatted according to the manufacturer's normal practice.

Two examples are

"5.3.006" and "1.12:2003-11-25-Standard."

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, VSTR. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the manufacturer's software version string.

VSTRn: Manufacturer's software version string.

Code assigned by the manufacturer for the installed version of the device's software.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.33 Device attributes—Device manufacturer's hardware version

DNP3 Object Library		Group:	0
		Variation:	243
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Device manufacturer's hardware version		

Description

This attribute is the version code of the manufacturer's device hardware.

The contents of this attribute is a free form string that is formatted according to the manufacturer's normal practice.

Two examples are

"Rev D" and "2004-122."

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, VSTR. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the manufacturer's hardware version string.

VSTRn: Manufacturer's hardware version string.

Code assigned by the manufacturer for the installed version of the device's hardware.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.34 Device attributes—User-assigned location name

DNP3 Object Library		Group:	0
		Variation:	245
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	User-assigned location name		

Description

This attribute is a name or code given to the location where the device is installed by the end user.

The content of this attribute is a free form string that is formatted according to the user's normal practice.

Three examples are

"1954 Broad Street," "Toora Wind Farm" and "Soho:Piccadilly Circus."

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, VSTR. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the user-assigned device location name or code.

VSTRn: User-assigned location name or code string.

Name or code for location where the device is installed. It is assigned by end user.

NOTE 1—It is recommended that device manufacturers provide a configurable means for the end user to locally store a text location name or code.

NOTE 2—If a user has a location naming convention consisting of sub-fields (like major and minor), it is recommended that he/she choose a delimiting character ("**:**," "~," "+," etc) as a standard method to separate the fields.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.35 Device attributes—User-assigned ID code/number

DNP3 Object Library		Group:	0
		Variation:	246
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	User-assigned ID code/number		

Description

This attribute is a code or number given to the device by the end user.

The content of this attribute is a free form string that is formatted according to the user's normal practice.

Two examples are

"4075" and "25-DS."

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, VSTR. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the user-assigned device code or number.

VSTRn: User-assigned device code or number string.

Code or number of device assigned by end user.

NOTE—It is recommended that device manufacturers provide a configurable means for the end user to locally store a text device code/number.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.36 Device attributes—User-assigned device name

DNP3 Object Library		Group:	0
		Variation:	247
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	User-assigned device name		

Description

This attribute is a name given to the device by the end user.

The content of this attribute is a free form string that is formatted according to the user's normal practice.

Two examples are

"Main Street Sub" and "500805RTU100."

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, VSTR. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the user-assigned device name.

VSTRn: User-assigned device name string.

Name of device assigned by end user.

NOTE—It is recommended that device manufacturers provide a configurable means for the end user to locally store a text device name.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.37 Device attributes—Device serial number

DNP3 Object Library		Group:	0
		Variation:	248
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Device serial number		

Description

This attribute is the serial number assigned by the device manufacturer.

The content of this attribute is a free form string that is formatted according to the manufacturer's normal practice.

Two examples are

"2003-07-04:1234" and "5Z4B-6xy9."

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, VSTR. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the manufacturer's serial number.

VSTRn: Manufacturer's serial number string.

Manufacturer's serial number.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.38 Device attributes—DNP3 subset and conformance

DNP3 Object Library		Group:	0
		Variation:	249
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	DNP3 subset and conformance		

Description

This attribute has three fields that identify the DNP3 subset level, a separator, and the DNP3 Test Procedure version for which the device was certified to conform to by an independent testing facility.

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, VSTR. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the subset level, colon, and test procedure versions.

VSTR1: Subset level.

There are four subset level values permitted:

ASCII “0” indicates unknown, not determined, or none claimed.

ASCII “1” indicates subset level 1.

ASCII “2” indicates subset level 2.

ASCII “3” indicates subset level 3.

VSTR1: Colon.

An ASCII colon, “**:**” (binary value 0x3A). This separator is mandatory and shall appear in the attribute string.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

VSTRn: Test procedure version.

DNP3 Test Procedure version for which the device was certified to conform to by an independent testing facility. This is usually a calendar year number such as "2003."

If the device has not been tested, this field shall be omitted from the object.

In the future, this field may contain more characters if conformance levels are coded differently.

NOTE—An example attribute has 6 characters and looks is coded as "1:2003".

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.39 Device attributes—Device manufacturer's product name and model

DNP3 Object Library		Group:	0
		Variation:	250
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Device manufacturer's product name and model		

Description

This attribute is the device manufacturer's product name and model.

The content of this attribute is a free form string that identifies an industry recognizable trade name and model. The string should not include the manufacturer's name as that appears in attribute variation 252.

Several examples are

> "Callisto," "D25 IED," "Form 5 Recloser," "SEL-351 Relay," "NTU-7500," "PDS Magna," "IntelliCap," and "Multicomm."

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, VSTR. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the manufacturer's product name and model.

VSTRn: Manufacturer's product name and model string.

Manufacture's product name and model.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.40 Device attributes—Device manufacturer's name

DNP3 Object Library		Group:	0
		Variation:	252
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Device manufacturer's name		

Description

This attribute is the name of the device manufacturer. An example is "123 SCADA, Ltd."

Coding

Pictorial

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, VSTR. (Refer to 11.1.)

UINT8: Length.

Specifies the number of octets in the manufacturer's name.

VSTRn: Manufacturer's name string.

Manufacture's name.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.41 Device attributes—Non-specific all attributes request

DNP3 Object Library		Group:	0
		Variation:	254
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	Non-specific all attributes request		

Description

This attribute is used as a shorthand to request an outstation to return all of its attributes in a single response. The master sends a single object header with this variation in lieu of including a possibly lengthy list of object headers in the request.

Coding

This variation does not have objects.

NOTE 1—This variation may only appear in a master request. It shall not be used in responses from outstations.

NOTE 2—Requests from a master use group 0, variation 254 qualifier code

— 0x00 where the start-stop range field indexes indicate from which set or sets of outstation attributes. Index 0 is the set of attributes defined by the DNP User's Group. Other indexes specify privately defined or vendor-specific attribute sets.

— 0x06.

NOTE 3—Devices that implement attributes are required to support this variation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.42 Device attributes—List of attribute variations

DNP3 Object Library		Group:	0
		Variation:	255
Group Name:	Device Attributes	Type:	Attrib
Variation Name:	List of attribute variations		

Description

This is a special attribute number that is used to retrieve a list of all of the device attribute variation numbers supported by the outstation at a specified index,[41] and the properties of those attributes. This object has a variable length that depends on the count of attribute variations supported by the outstation.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
								Attribute data type code
								Length
								1st attribute variation
								1st attribute properties
								2nd attribute variation number
								2nd attribute properties
								●●●
								Last attribute variation number
								Last attribute's properties

Formal structure

UINT8: Attribute data type code.

Specifies the attribute data type code, U8BS8LIST or U8BS8EXLIST. (Refer to 11.1.)

UINT8: Length

Specifies the number of octets in the following list of variation numbers and properties (used in conjunction with the preceding field to allow more than 255 octets in the list).

SET of n

{

UINT8: Attribute variation number.

The number of an attribute variation that is supported in the device.

[41] The different sets of attributes are retrieved using the "index" field in the DNP3 application header to denote the set number —index 0 being used for the attribute set defined by the DNP User Group.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

BSTR8: Attribute properties.

Bit 0 indicates whether the device attribute is writable by the master. If the bit is set, the master can change the respective attribute by sending a request message having function code [WRITE].

Bits 1 through 7 are reserved.

}

where n is the number of device attribute variations supported by the outstation.

NOTE 1—The list does not include variations 254 and 255.

NOTE 2—Devices that implement attributes are required to support this variation.

NOTE 3—This attribute variation is not writable.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.43 Binary input—Packed format

DNP3 Object Library		Group:	1
		Variation:	1
Group Name:	Binary Input	Type:	Static
Variation Name:	Packed format		

Description

Object group 1, variation 1 is used to report the current value of a binary input point. See 11.1 for a description of a Binary Input Point Type.

Variation 1 objects contain a single-bit binary input state without status flags.

Coding

This object is coded as a single bit, with a possible value of 0 or 1, for each point index. When multiple points are contained in a single response message, they are packed, as illustrated in the following pictorial.

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
m+7	m+6	m+5	m+4	m+3	m+2	m+1	m	States
:	:	:	:	:	:	:	:	
0	0	0	n	n-1	n-2	n-3	n-4	

This figure depicts the bit-packing sequence for a response containing bit indexes m through n. As can be seen, the first bit is in the bit 0 position of the first octet, the second bit is in the bit 1 position of the first octet, etc. Any unused bits in the final octet are returned as 0.

Formal structure

BSTRn: States

n in BSTRn represents the number of indexes reported in the bit string.

Bit 0 in the bit string corresponds to the lowest point index reported. Subsequent bits correspond to sequentially increasing point indexes.

Unused bits in the last octet are padded with 0.

Each bit has a value of 0 or 1, representing the state of the physical or logical input.

NOTE—This variation does not contain flag-type information. Data returned in this variation is assumed to be *on-line* with no failure indications. Variation 2, Binary Input with Flags, should be used for points that have an abnormal condition that can be reported with flag bits.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.44 Binary input—With flags

DNP3 Object Library		Group:	1
		Variation:	2
Group Name:	Binary Input	Type:	Static
Variation Name:	With flags		

Description

Object group 1, variation 2 is used to report the current value of a binary input point. See 11.1 for a description of a Binary Input Point Type.

Variation 2 objects contain a status octet that includes the state of the binary input.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
ST	0	CF	LF	RF	CL	RS	OL	State and flag octet

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	CHATTER_FILTER
Bit 6:	Reserved, always 0
Bit 7:	STATE—Has a value of 0 or 1, representing the state of the physical or logical input.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.45 Binary input event—Without time

DNP3 Object Library		Group:	2
		Variation:	1
Group Name:	Binary Input Event	Type:	Event
Variation Name:	Without time		

Description

Object group 2, variation 1 is used to report events related to a binary input point. See 11.1 for a description of a Binary Input Point type.

Variation 1 objects contain an octet for reporting the state of the input and status flags.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
ST	0	CF	LF	RF	CL	RS	OL	State and flag octet

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	CHATTER_FILTER
Bit 6:	Reserved, always 0
Bit 7:	STATE—Has a value of 0 or 1, representing the state of the physical or logical input.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.46 Binary input event—With absolute time

DNP3 Object Library		Group:	2
		Variation:	2
Group Name:	Binary Input Event	Type:	Event
Variation Name:	With absolute time		

Description

Object group 2, variation 2 is used to report events related to a binary input point. See 11.1 for a description of a Binary Input Point Type.

Variation 2 objects contain an octet for reporting the state of the input and status flags, and the absolute time when the event occurred.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	ST	0	CF	LF	RF	CL	RS	OL		State and flag octet
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	CHATTER_FILTER
Bit 6:	Reserved, always 0
Bit 7:	STATE—Has a value of 0 or 1, representing the state of the physical or logical input.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.47 Binary input event—With relative time

DNP3 Object Library		Group:	2
		Variation:	3
Group Name:	Binary Input Event	Type:	Event
Variation Name:	With relative time		

Description

Object group 2, variation 3 is used to report events related to a Binary Input point. See 11.1 for a description of a Binary Input Point Type.

Variation 3 objects contain an octet for reporting the state of the input and status flags, and the relative time when the event occurred. A preceding common time-of-occurrence (CTO) object, group 51, establishes the basis of the relative time.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	ST	0	CF	LF	RF	CL	RS	OL		State and flag octet
									b0	Relative time-of-occurrence
b15										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	CHATTER_FILTER
Bit 6:	Reserved, always 0
Bit 7:	STATE—Has a value of 0 or 1, representing the state of the physical or logical input.

UINT16: Relative time-of-occurrence

Time, in milliseconds, when the event occurred, with respect to the preceding common time-of-occurrence object.

NOTE—See 11.1 for flag bit descriptions. See the CTO description, group 51, for more information about common time-of-occurrence and relative times.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.48 Double-bit binary input—Packed format

DNP3 Object Library		Group:	3
		Variation:	1
Group Name:	Double-bit Binary Input	Type:	Static
Variation Name:	Packed format		

Description

Object group 3, variation 1 is used to report the current value of a double-bit binary input point. See 11.1 for a description of a Double-bit Binary Input Point Type.

Variation 1 objects contain double-bit binary input states packed without status flags.

Coding

This object is coded as a two-bit, unsigned integer with possible values of 0 to 3, for each point index. When multiple points are contained in a single response message, they are packed, as illustrated in the following pictorial.

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position	
m+3		m+2		m+1		m		b0	Double-bit states
m+7		m+6		m+5		m+4			
:	:	:	:	:	:	:	:	:	
0		n		n-1		n-2			

This figure depicts the bit-packing sequence for a response containing double-bit group indexes m through n. As can be seen, the first double-bit state is in the bit 0 and 1 position of the first octet, the second double-bit state is in the bit 2 and 3 position of the first octet, etc. Any unused double-bit groups in the final octet are returned as 0.

Formal structure

SET of n UINT2: State

n represents the number of indexes reported.

The first index is aligned on bit 0 of the first octet.

Unused bits in the last octet are padded with 0.

Each integer contains a state as defined for a Double-bit Binary Input Point Type in 11.1. These values represent states INTERMEDIATE, DETERMINED_OFF, DETERMINED_ON, and INDETERMINATE.

NOTE—This variation does not contain flag-type information. Data returned in this variation is assumed to be *on-line* with no failure indications. Variation 2, Double-bit Binary Input with flags, should be used for points that have an abnormal condition that can be reported with flag bits.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.49 Double-bit binary input—With flags

DNP3 Object Library		Group:	3
		Variation:	2
Group Name:	Double-bit Binary Input	Type:	Static
Variation Name:	With flags		

Description

Object group 3, variation 2 is used to report the current state of a double-bit binary input point. See 11.1 for a description of a Double-bit Binary Input Point Type.

Variation 2 objects contains a status octet that includes the state of the double-bit binary input.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
ST		CF	LF	RF	CL	RS	OL	State and flag octet

Formal structure

BSTR6: Flags

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	CHATTER_FILTER

UINT2: State

This integer contains the state of the double-bit binary input as defined for a Double-bit Binary Input Point Type in 11.1. These values represent states INTERMEDIATE, DETERMINED_OFF, DETERMINED_ON, and INDETERMINATE

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.50 Double-bit binary input event—Without time

DNP3 Object Library		Group:	4
		Variation:	1
Group Name:	Double-bit Binary Input Event	Type:	Event
Variation Name:	Without time		

Description

Object group 4, variation 1 is used to report events related to a double-bit binary input point. See 11.1 for a description of a Double-bit Binary Input Point Type.

Variation 1 objects contains a status octet that includes the input state.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
State		CF	LF	RF	CL	RS	OL	State and flag octet

Formal structure

BSTR6: Flags

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	CHATTER_FILTER

UINT2: State

Each integer contains a state as defined for a Double-bit Binary Input Point Type in 11.1. These values represent states INTERMEDIATE, DETERMINED_OFF, DETERMINED ON, and INDETERMINATE

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.51 Double-bit binary input event—With absolute time

DNP3 Object Library		Group:	4
		Variation:	2
Group Name:	Double-bit Binary Input Event	Type:	Event
Variation Name:	With absolute time		

Description

Object group 4, variation 2 is used to report events related to a double-bit binary input point. See 11.1 for a description of a Double-bit Binary Input Point Type.

Variation 2 objects contain a status octet for reporting the state of the input and status flags, and a time-of-occurrence.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	State		CF	LF	RF	CL	RS	OL		State and flag octet
									b0	
										Time-of-occurrence
B47										

Formal structure

BSTR6: Flags

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	CHATTER_FILTER

UINT2: State

This integer contains a state as defined for a Double-bit Binary Input Point Type in 11.1. These values represent states INTERMEDIATE, DETERMINED_OFF, DETERMINED_ON, and INDETERMINATE

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.52 Double-bit binary input event—With relative time

DNP3 Object Library		Group:	4
		Variation:	3
Group Name:	Double-bit Binary Input Event	Type:	Event
Variation Name:	With relative time		

Description

Object group 4, variation 3 is used to report events related to a double-bit binary input point. See 11.1 for a description of a Double-bit Binary Input Point Type.

Variation 3 objects contain a status octet for reporting the state of the input and status flags, and a relative time-of-occurrence. A preceding common time-of-occurrence (CTO) object, group 51, establishes the basis of the relative time.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	ST		CF	LF	RF	CL	RS	OL		Flag octet
									b0	Relative time-of-occurrence
B15										

Formal structure

BSTR6: Flags

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	CHATTER_FILTER

UINT2: State

This integer contains a state as defined for a Double-bit Binary Input Point Type in 11.1. These values represent states INTERMEDIATE, DETERMINED_OFF, DETERMINED_ON, and INDETERMINATE

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT16: Relative time-of-occurrence

Time, in milliseconds, when the event occurred, relative to the preceding common time-of-occurrence object.

NOTE—See 11.1 for flag bit descriptions. See the CTO description, group 51, for more information on common time-of-occurrence and relative times.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.53 Binary output—Packed format

DNP3 Object Library		Group:	10
		Variation:	1
Group Name:	Binary Output	Type:	Static
Variation Name:	Packed format		

Description

Object group 10, variation 1 is used to control or report the state of one or more binary output points. Subclause 11.1 provides a description of the Binary Output Point Type.

This object is suitable for DNP3 devices with electro-mechanical relays, pseudo-controls, and other output mechanisms.

Control actions are initiated by sending a request message from the master with the Application Layer function code ***write*** (2). This variation should only be used to control points that implement a complementary latch model.

To obtain the output state of a binary output point, regardless of which point type model is implemented at the respective index, a request is sent from the master with Application Layer function code ***read*** (1). However, the state returned is only meaningful if the characteristics of the point are such that the output retains a meaningful state for a significant amount of time after a control command is issued to the point. The state is readable even if the binary output point is controlled through some other means, such as a manual operation, software automation command, or the master issuing a control command with object g12v1.

Coding

This object is coded as a single bit, with a possible value of 1 or 0, for each point index. When multiple points are contained in a single message, they are packed, as illustrated in the following pictorial.

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
m+7	m+6	m+5	m+4	m+3	m+2	m+1	m	
.	.	.	.	.	.	.	.	Output states
0	0	0	n	n-1	n-2	n-3	n-4	

This figure depicts the bit-packing sequence for a command containing bit indexes m through n. As can be seen, the first bit is in the bit 0 position of the first octet, the second bit is in the bit 1 position of the first octet, etc. Any unused bits in the final octet are set to 0.

Formal structure

BSTRn: Output states

n in BSTRn represents the number of indexes appearing in the bit string.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Bit 0 in the bit string corresponds to the lowest point index to be commanded. Subsequent bits correspond to sequentially increasing point indexes.

Bit 0 in the bit string is aligned with bit 0 of the first octet.

Unused bits in the last octet are padded with 0.

Each bit has a value of 1 or 0 where 1 indicates the output is active and 0 indicates the output is not active. When the output state is not meaningful, the bit shall be 0.

NOTE 1—The preferred method for issuing control commands to a binary output point is by including a g12v1 object with the ***select*** and ***operate*** function codes. The reason is that the status of the operation is returned in the command response, whereas the response to a ***write*** request message does not describe the success of the operation.

NOTE 2—A point index in object group 10 corresponds to the same physical or logical point as the identical index in object groups 11, 12, and 13.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.54 Binary output—Output status with flags

DNP3 Object Library		Group:	10
		Variation:	2
Group Name:	Binary Output	Type:	Static
Variation Name:	Output status with flags		

Description

Object group 10, variation 2 is used to report the status of a binary output point. Subclause 11.1 provides a description of the Binary Output Point Type.

Variation 2 objects contains a flag octet that includes the activation state of the continuous output signal.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
ST	0	0	LF	RF	CL	RS	OL	Flag octet

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	Reserved, always 0
Bit 6:	Reserved, always 0
Bit 7:	STATE—Value is 1 or 0, where 1 indicates that the output signal is active and 0 indicates that the output signal is not active. When the output state is not meaningful, the STATE flag shall be 0.

NOTE 1—See 11.1 for flag bit descriptions.

NOTE 2—A point index in object group 10 corresponds to the same physical or logical point as the identical index in object groups 11, 12, and 13.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.55 Binary output event—Status without time

DNP3 Object Library		Group:	11
		Variation:	1
Group Name:	Binary Output Event	Type:	Event
Variation Name:	Status without time		

Description

A Binary Output Event object is an instance of a report for an outstation's corresponding Binary Output Static object (object group 10). Subclause 11.1 provides a description of the Binary Output Point Type. A Binary Output Event may be generated for a point only if the outstation returns an output static object for that point in response to a static (Class 0) request. A change event may be generated by an outstation when it detects something of interest has occurred. Examples of this include a change in output point state, or a change in output object flags. A binary output event object may also be generated due to other application dependent reasons.

For binary output points, changes to the exception flags, and changes of the output state **when the underlying point retains a value for a significant amount of time**, are reported with Binary Output Event objects. The binary output point's status (including flags) at the time when the event is generated and queued is the status placed into the Binary Output Event object.

This object shall not be generated to simply report that a command was received. Object Group 13 objects should be used for this purpose.

Generation of Binary Output Events is optional for an outstation, and may occur in one or more of the following situations:

— Output point state changed—Upon a control from a master or upstream device that results in a change to the status of an output point, the outstation may generate an event object for that point. For example, where multiple masters control the same output point on an outstation, a change event may be generated to one or more master stations to indicate that the output changed state.

— External effects—Mechanisms other than a control from a master may result in a change to the status of a point represented by an output object. An event may be generated to indicate to the master that the output point status has changed. For example, a local automation sequence may override the state of a point set by a previous control from the master.

— Flags changed—Where an outstation generates Binary Output Events for a particular point, an event shall be generated when any of the flags for the output point change. Note that the flags in the flags octet relates to the output point status and not to the status value of any related input point(s).

Outstations that implement this object:

— Shall be able to configure whether output change events are generated on output points.

— Should support Application Layer Function Code 22 (ASSIGN_CLASS) for Object Group 10, Variation 0, if the outstation device supports Assign Class functions on other objects.

When events are configured for a particular output point, the outstation:

— May generate an event when something of interest happens on the output point.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— When flags are supported for the point, shall generate an event when any of the status flags for the output point change.

— Shall generate an event upon a change in the output point state, where the **output point retains a meaningful state for a significant amount of time**. A Binary Output Event shall be reported for transitions both to the active and to the not active output states.

— Shall not generate an output change event solely for the purpose of indicating that a command has occurred on a point. An Object Group 13 object may be generated to indicate this. Object Group 11 objects are independent from Object Group 13 objects.

Variation 1 objects contain a flag octet that indicates the status of the output point. For output points that retain a state, the state shall be indicated in the flag octet. Where a state is reported in a Binary Output Event, the state shall be that of the output point itself, not feedback resulting from the plant being driven, and any such change shall generate an event.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
ST	0	0	LF	RF	CL	RS	OL	Flag octet

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	Reserved, always 0
Bit 6:	Reserved, always 0
Bit 7:	STATE—Value is 1 or 0, where 1 indicates that the output signal is active and 0 indicates that the output signal is not active. Where the output object does not have a meaningful output state, the STATE flag shall be 0.

NOTE 1—A point index in object group 11corresponds to the same physical or logical point as the identical index in object groups 10, 12, and 13.

NOTE 2—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 10 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.56 Binary output event—Status with time

DNP3 Object Library		Group:	11
		Variation:	2
Group Name:	Binary Output Event	Type:	Event
Variation Name:	Status with time		

Description

See object group 11, variation 1 for a description of Binary Output Events, event detection, and implementation requirements.

Variation 2 objects contain a flag octet that indicates the status of the output point and a time-of-occurrence.

Coding

Pictorial

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	Reserved, always 0
Bit 6:	Reserved, always 0
Bit 7:	STATE—Value is 1 or 0, where 1 indicates that the output signal is active and 0 indicates that the output signal is not active. Where the output object does not have a meaningful output state, the STATE flag shall be 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—A point index in object group 11 corresponds to the same physical or logical point as the identical index in object groups 10, 12, and 13.

NOTE 2—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 10 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.57 Binary output command—Control relay output block—also known as CROB

DNP3 Object Library		Group:	12
		Variation:	1
Group Name:	Binary Output Command	Type:	Cmnd
Variation Name:	Control relay output block—also known as CROB		

Description

Object group 12, variation 1 is used to perform digital control operations at binary output points. It is suitable for DNP3 devices with electro-mechanical relays, pseudo-controls, and other output mechanisms. Each g12v1 object contains a block of information that applies to a single index.

Subclause 11.1 provides a description of the Binary Output Point Type. In summary, there are three control models for which this object is appropriate.

a) Activation model—Has a single virtual or physical output. Its purpose is to initiate an action (i.e., "cause it to happen").

b) Complementary latch model—Has a single virtual or physical output that remains latched in an active or non-active state depending on which command is received.

c) Complementary two-output model—Has two virtual or physical outputs, named close and trip. One or the other output is set active momentarily depending on which command is received.

Control actions are initiated by sending one or more request messages from a master with an appropriate Application Layer function code, ***select*** (3), ***operate*** (4), ***direct operate*** (5), or ***direct operate without acknowledgment*** (6), along with a g12v1, object for each point index to be controlled. This object shall not be used with a ***write*** function code (2).

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	TCC		CR	QU	Op Type					Control code
b7									b0	Count
									b0	
										On-time
b31										
									b0	
										Off-time
b31										
	RES	Status code								Status code & Reserved bit

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Formal structure

UINT4: Control code, Operation Type field [Op Type]

This field is used in conjunction with the *TCC* field to specify a control operation. See operational functions in the notes that follow for additional details. The code names are

0:	NUL
1:	PULSE_ON
2:	PULSE_OFF
3:	LATCH_ON
4:	LATCH_OFF
5 to 15:	Undefined

BSTR1: Control code, Queue field [QU]

This field is obsolete. The master shall always set this bit to 0. Outstations that receive a g12v1 object with this bit set shall return a status code NOT_SUPPORTED in the response.

BSTR1: Control code, Clear field [CR]

Support for this field is optional. If the device supports commands with the clear bit set:

— It shall support the case where the Op Type code is NUL, and may support the command for other Op Type values.

— When the clear bit is set, the device shall remove pending control commands for that index and stop any control operation that is in progress for that index. The indexed point shall go to the state that it would have if the command were allowed to complete normally. If a non-NUL Op Type code is requested, the new command shall be initiated immediately after the cancellation actions complete.

When the clear bit is set and the *TCC* - *Op Type* combination is not supported, the device shall return status code NOT_SUPPORTED in the response.

UINT2: Control code, Trip-Close Code field [TCC]

This field is used in conjunction with the *Op Type* field to specify a control operation. See operational functions for additional details. The code names are

0:	NUL
1:	CLOSE
2:	TRIP
3:	RESERVED

UINT8: Count.

This is the number of times the outstation shall execute the operation. Counts greater than 1 generate a series of pulses or repeated operations for the point. Both *On-time* and *Off-time* values are obeyed as illustrated in the figures under timing illustrations, subject to the comments regarding timing in interpreting the time fields.

Implementation of a zero-count functionality is optional. A count value of 0 indicates that the output operation shall not be executed. Setting the count value to 0 is a useful

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

technique for testing communications without affecting an output. When the outstation receives a 0 value, it shall:

— Not change the output.

— Ignore the On-time and Off-time values.

— Return the same status code as if the execution had been attempted.

An outstation shall return status code NOT_SUPPORTED in the response when the count value is 0 in the request and the outstation does not implement the zero-count functionality.

UINT32: On-time.

This is the duration, expressed as the number of milliseconds, that the output drive remains active. See interpreting the time fields for more details.

UINT32: Off-time.

This is the duration, expressed as the number of milliseconds that the output drive remains non-active. See interpreting the time fields for more details.

UINT7: Status code.

This value shall be set to 0 in request messages.

In response messages, this value represents the status of the selected or executed command. An enumeration of status codes appears in 11.1.

BSTR1: Reserved [RES].

This bit is reserved. The master and outstation shall always set it to 0.

Notes

Timing Illustrations

This section illustrates just two of the many timing possibilities that can appear in a g12v1 object.

PULSE ON with Count = 3

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

PULSE OFF with Count = 2 (Not interoperable)

Operational Functions

Interoperable commands

Only a few of the 64 possible *TCC* and *Op Type* field bit combinations are interoperable. Table A.1 indicates, for each point index, which commands are optional, preferred, not permitted, or not interoperable, depending on which point model is implemented.

Table A.1—Interoperable control commands

TCC field	Op Type field	Point model in outstation	Support requirements
NUL	NUL	All	Optional
NUL	PULSE_ON	Activation	May support
		Complementary latch	Not permitted
		Complementary two-output	Not permitted
NUL	LATCH_ON	Activation	May support
		Complementary latch	Preferred support
		Complementary two-output	May support
NUL	LATCH_OFF	Activation	May support
		Complementary latch	Preferred support
		Complementary two-output	May support
CLOSE	PULSE_ON	Activation	May support
		Complementary latch	May support
		Complementary two-output	Preferred support
TRIP	PULSE_ON	Activation	May support
		Complementary latch	May support
		Complementary two-output	Preferred support
All other combinations		All	Not interoperable

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Table A.2 indicates for a single point index, what action the outstation performs based on the contents of the interoperable control codes and the point model implemented.

Table A.2—Actions performed by outstation for interoperable commands

Row #	Control Code	TCC Field	Op Type Field	Clear Field	Action
1	0x00	NUL	NUL	0	Does not initiate an action or change an in-progress or pending command. Values in *On-time* and *Off-time* fields are ignored.
2	0x20	NUL	NUL	1	Cancel in-progress and pending commands. Values in *On-time* and *Off-time* fields are ignored.
3	0x01	NUL	PULSE_ON	0	For activation model, set output to active for the duration of the *On-time*. For both complementary models, return NOT_SUPPORTED status.
4	0x21	NUL	PULSE_ON	1	For activation model, cancel in-progress and pending commands and then set output to active for the duration of the *On-time*. For both complementary models, return NOT_SUPPORTED status.
5	0x03	NUL	LATCH_ON	0	For activation model, set output to active for the duration of the *On-time*. For complementary latch model, set the output to active. For complementary two-output model, set the close output to active for the duration of the *On-time*.
6	0x23	NUL	LATCH_ON	1	Cancel in-progress and pending commands. Afterwards, initiate the action specified in row 5.
7	0x04	NUL	LATCH_OFF	0	For activation model, set output to active for the duration of the *On-time*. For complementary latch model, set the output to inactive. For complementary two-output model, set the trip output to active for the duration of the *On-time*.
8	0x24	NUL	LATCH_OFF	1	Cancel in-progress and pending commands. Afterwards, initiate the action specified in row 7.
9	0x41	CLOSE	PULSE_ON	0	For activation model, set output to active for the duration of the *On-time*. For complementary latch model, set the output to active. For complementary two-output model, set the close output to active for the duration of the *On-time*.
10	0x61	CLOSE	PULSE_ON	1	Cancel in-progress and pending commands. Afterwards, initiate the action specified in row 9.
11	0x81	TRIP	PULSE_ON	0	For activation model, set output to active for the duration of the *On-time*. For complementary latch model, set the output to inactive. For complementary two-output model, set the trip output to active for the duration of the *On-time*.
12	0xA1	TRIP	PULSE_ON	1	Cancel in-progress and pending commands. Afterwards, initiate the action specified in row 11.

Additional Requirements

Vendors of outstation devices are responsible for assigning control codes that are appropriate to their device. For example, a manufacturer might assign CLOSE – PULSE_ON and TRIP – PULSE_ON to a breaker and NUL – LATCH_ON and NUL – LATCH_OFF to a pseudo point.

Each index point shall implement only one of the control models. It is acceptable for vendors to provide configuration to select a model, but during operation, the point type shall remain fixed. Differing point indexes may support dissimilar point type models; that is, not all the indexes within a device need to have the same model.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

For each point index implementing the activation model, at least one of the *TCC – Op Type* field combinations marked as "May support" in Table A.1 shall be accepted for this point type. If more than one is accepted, then each shall perform the identical function.

For each point index implementing one of the complementary models, at least one of the following command pairs shall be chosen:

NUL – LATCH_ON ↔ NUL – LATCH_OFF

or

CLOSE – PULSE_ON ↔ TRIP – PULSE_ON

For each point index implementing one of the complementary models, if both command pairs are supported, then

NUL – LATCH_ON and CLOSE – PULSE_ON shall perform the identical function

and

NUL – LATCH_OFF and TRIP – PULSE_ON shall perform the identical function

Vendors are free to implement operations using the PULSE_OFF value in the *Op Type* field, but these may not be interoperable with all DNP3 masters, and alternate outstation vendors may not implement these operations. One use would be the generation of a pulse train that ends in the output remaining active. For example, if the output value was active prior to a PULSE_OFF command, then the output would go inactive for the time determined by the *Off-time* and then the output would change to active.

It is acceptable to use two separate indexes, each implementing the activation model, to provide complementary functionality. For example, one index can drive the "trip" coil and another index can drive the "close" coil of a single circuit breaker.

Outstation vendors may choose one of the following procedures when a command is received for a presently operating point:

- Wait for completion of the previous command before beginning the new command.
- Return an error status code to the master.
- Terminate the presently executing command and initiate the new command.

Interpreting the Time Fields

The values in the *On-time* and *Off-time* fields are *suggestions* for time durations. A few common interpretations are as follows:

- Always ignore. The outstation uses configured values in lieu of the time values in the object. This choice minimizes the amount of configuration in the master.
- Use unless zero. The outstation uses the values supplied unless they are zero. A 0-time value indicates that the device shall substitute its pre-configured value. This choice maximizes output flexibility but allows masters to execute "unwise" commands (such as "pulse for 50 days").

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— Use if reasonable. The outstation uses the values if they are within an acceptable or configured range. Otherwise, the outstation substitutes an appropriate value. This choice provides limited output flexibility but increases the safety of the operation.

— A combination of the above.

Master configuration

A master device shall be configurable as to which control codes it shall issue on a per-index basis. The choices shall include all of the odd-numbered rows, beginning with row 3, in Table A.2. In addition, the master shall allow configuration of the *On-time* field. It may optionally provide configurable *Off-time* and *Count* fields. If these are not configurable, the *Off-time* should be 0 and the *Count* should be 1.

Minimal Outstation Implementation

At each binary output point index, the outstation shall choose a point type model and which of the *TCC* and *Op Type* field values it shall support. In addition, it shall support a *Count* value of 1 and an *Off-time* value of 0.

Point Index Correlation

A point index in object group 12 corresponds to the same physical or logical point as the identical index in object groups 10, 11 and 13.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.58 Binary output command—Pattern control block—also known as PCB

DNP3 Object Library		Group:	12
		Variation:	2
Group Name:	Binary Output Command	Type:	Cmnd
Variation Name:	Pattern control block—also known as PCB		

Description

Object group 12, variation 2 is used to control multiple binary output points. It is suitable for DNP3 devices with electro-mechanical relays, pseudo-controls, and other output mechanisms. This variation is intended for performing control operations at multiple points simultaneously where control actions are accomplished concurrently.

Subclause 11.1 provides a description of the Binary Output Point Type. In summary, there are three control models for which this object is appropriate.

a) Activation model—Has a single virtual or physical output. Its purpose is to initiate an action (i.e., "cause it to happen").

b) Complementary latch model—Has a single virtual or physical output that remains latched in an active or non-active state depending on which command is received.

c) Complementary two-output model—Has two virtual or physical outputs, named close and trip. One or the other output is set active momentarily depending on which command is received.

Executing controls is initiated by sending one or more request messages from a master with an appropriate Application Layer function code, ***select*** (3), ***operate*** (4), ***direct operate*** (5), or ***direct operate without acknowledgment*** (6), along with a single g12v2 object followed by one or more Pattern Mask objects, g12v3. This object may not be used with a ***write*** function code, 2.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	TCC		CR	QU	Op Type					Control code
b7									b0	Count
									b0	
										On-time
b31										
									b0	
										Off-time
b31										
	RES	Status code								Status code & Reserved bit

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Formal structure

Please see the formal structure for object g12v1, which is identical.

NOTE—Please see notes for object g12v1, which are identical.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.59 Binary output command—Pattern mask

DNP3 Object Library		Group:	12
		Variation:	3
Group Name:	Binary Output Command	Type:	Cmnd
Variation Name:	Pattern mask		

Description

Object group 12, variation 3 is used to control multiple binary output points.

This variation is used in conjunction with an object g12v2 to specify which specific points are to be simultaneously controlled. Please see the g12v2 insert sheet.

Variation 3 objects contain a single bit for each output point index.

Coding

This object is coded as a bit string where each bit corresponds to a point index. Indexes are packed sequentially.

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
m+7	m+6	m+5	m+4	m+3	m+2	m+1	m	
.	.	.	.	.	.	.	.	Included point indexes
0	0	0	n	n-1	n-2	n-3	n-4	

This figure depicts the bit-packing sequence for an object containing bit indexes m through n. As can be seen, the first bit is in the bit 0 position of the first octet, the second bit is in the bit 1 position of the first octet, etc. Any unused bits in the final octet are set to 0.

Formal structure

BSTRn: Included point indexes

n in BSTRn represents the number of indexes appearing in the bit string.

Bit 0 in the bit string corresponds to the lowest point index. Subsequent bits correspond to sequentially increasing point indexes.

Bit 0 in the bit string is aligned with bit 0 of the first octet.

Unused bits in the last octet are padded with 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Each bit has a value of 1 or 0, where 1 indicates that the respective point shall participate in the concurrent signal activation specified by the preceding Pattern Control Block object, and 0 indicates that the point index shall not participate.

NOTE—A point index in object group 12 corresponds to the same physical or logical point as the identical index in object groups 10, 11, and 13.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.60 Binary output command event—Command status without time

DNP3 Object Library		Group:	13
		Variation:	1
Group Name:	Binary Output Command Event	Type:	Event
Variation Name:	Command status without time		

Description

A Binary Output Command Event object reports that a command has been attempted on an outstation's corresponding binary output point. Example usages of this object include:

— Reporting to a master device that an outstation has received a control from another master

— Reporting to a master device that a successful or unsuccessful control was attempted

— Reporting to a master device that a control request was made by an internal application

— Reporting to a master device that a control was received and processed at an outstation when issued through a non-originating device (e.g., Data Concentrator)

A command event may be generated by an outstation when it receives a control command for an output point from any internal or external source. Examples of this include a command request received though DNP3 containing a CROB, a command request from a different protocol, a control change command from a local automation application.

Where the controlled binary output point retains an output state, a control request to the output point would include state information. That requested control state is returned in this binary output command event. This object does not return state or status information for the output point itself or any feedback input associated with the control. Output command event objects received by a master are intended to be used for informational purposes, such as being logged.

Generation of Binary Output Command Events is optional for an outstation. This object may be used in conjunction with Binary Output Change Event objects.

Outstations that implement this object:

— Shall be able to configure whether output command events are generated on output points.

— Should support Application Layer Function Code 22 (ASSIGN_CLASS) for object group 12, Variation 0 if the outstation device supports Assign Class functions on other objects.

— May generate events when controls are commanded from internal sources (e.g., application), external sources requesting controls (e.g., protocol control request), or both.

— Shall not generate this event until a command is fully processed. For example when receiving a Select Before Operate control request, an event shall not be generated following reception of a Select command until either the Select fails or the Operate is processed.

— Shall not generate this event as a result of an output point state change or a status change. Object group 11 objects are used for this purpose. Object group 13 objects are independent from object group 11 objects.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— Should provide the result of processing the control request in the control status field of this object. Where this status can be provided, it shall be in the same format as the control status field provided in DNP3 object 12, variation 1.

— Should provide at least an indication of whether the processed control request was accepted (status = 0) or not accepted for unknown reasons (status = 127). Where possible, it is recommended that the outstation return known, relevant status codes. Where the processed control was received through a DNP3 request, for example, the status code returned in the output command event status code should be the same status code used in the response to the control request.

— May choose to send output command events (1) only when the control actually succeeds (i.e., status code = 0), or (2) for any control command regardless of whether it succeeds or fails (i.e., status code >= 0)

Variation 1 objects contain information regarding the commanded state of the output (where the point retains state information) and status information indicating the status that resulted when the outstation processed the requested control on the point. Information provided in this object relates to a control request, and not to the resultant state or status of the point controlled.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	CS	Status code								State and status

Formal structure

UINT7: Status Code

This value represents the status from processing the command that this event object is reporting. An enumeration of status codes appears in 11.1.

BSTR1: Commanded State

The commanded flag has a value of 0 or 1, representing the control requested for the physical or logical output. 0 = Latch Off / Trip / NULL, 1 = Latch On / Close. Where the commanded state is unknown, the commanded state flag shall be 0.

NOTE 1—A point index in object group 13 corresponds to the same physical or logical point as the identical index in object groups 10, 11, and 12.

NOTE 2—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 12 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.61 Binary output command event—Command status with time

DNP3 Object Library		Group:	13
		Variation:	2
Group Name:	Binary Output Command Event	Type:	Event
Variation Name:	Command status with time		

Description

See object g13v1 for a description of Binary Output Command Events and event detection.

Variation 2 objects contain a status octet that indicates the commanded state from the control request, the status code that resulted when the outstation processed the requested control, and a time-of-occurrence. The time-of-occurrence represents the time at which the control request was processed. Information provided in this object relates to a control request, and not to the resultant state or status of the point controlled.

Coding

Pictorial

Formal structure

UINT7: Status Code

This value represents the status from processing the command that this event object is reporting. An enumeration of status codes appears in 11.1.

BSTR1: Commanded State

The commanded flag has a value of 0 or 1, representing the control requested for the physical or logical output. 0 = Latch Off / Trip / NULL, 1 = Latch On / Close. Where the commanded state is unknown, the commanded state flag shall be 0.

UINT48: Time-of-occurrence

Time when the control was processed, expressed as the number of milliseconds since the start of January 1, 1970.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

NOTE 1—A point index in object group 13 corresponds to the same physical or logical point as the identical index in object groups 10, 11, and 12.

NOTE 2—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 12 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.62 Counter—32-bit with flag

DNP3 Object Library		Group:	20
		Variation:	1
Group Name:	Counter	Type:	Static
Variation Name:	32-bit with flag		

Description

Object group 20, variation 1 is used to report the current value of a counter point. See 11.1 for a description of a Counter Point Type.

Variation 1 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	DC	RO	LF	RF	CL	RS	OL		Flag octet
									b0	
										Count value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	DISCONTINUITY
Bit 7:	Reserved, always 0.

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.63 Counter—16-bit with flag

DNP3 Object Library		Group:	20
		Variation:	2
Group Name:	Counter	Type:	Static
Variation Name:	16-bit with flag		

Description

Object group 20, variation 2 is used to report the current value of a counter point. See 11.1 for a description of a Counter Point Type.

Variation 2 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	DC	RO	LF	RF	CL	RS	OL		Flag octet
									b0	Count value
b15										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	DISCONTINUITY
Bit 7:	Reserved, always 0.

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.64 Counter—32-bit with flag, delta

DNP3 Object Library		Group:	20
		Variation:	3
Group Name:	Counter	Type:	Static
Variation Name:	32-bit with flag, delta		

Description

Object group 20, variation 3 is used to report the current value of a delta counter point.

Variation 3 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	RO	LF	RF	CL	RS	OL		Flag octet
									b0	
										Count value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	Reserved, always 0.
Bit 7:	Reserved, always 0.

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.65 Counter—16-bit with flag, delta

DNP3 Object Library		Group:	20
		Variation:	4
Group Name:	Counter	Type:	Static
Variation Name:	16-bit with flag, delta		

Description

Object group 20, variation 4 is used to report the current value of a delta counter point.

Variation 4 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	RO	LF	RF	CL	RS	OL		Flag octet
									b0	Count value
b15										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	Reserved, always 0.
Bit 7:	Reserved, always 0.

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.66 Counter—32-bit without flag

DNP3 Object Library		Group:	20
		Variation:	5
Group Name:	Counter	Type:	Static
Variation Name:	32-bit without flag		

Description

Object group 20, variation 5 is used to report the current value of a counter point. See 11.1 for a description of a Counter Point Type.

Variation 5 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

Formal structure

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.67 Counter—16-bit without flag

DNP3 Object Library		Group:	20
		Variation:	6
Group Name:	Counter	Type:	Static
Variation Name:	16-bit without flag		

Description

Object group 20, variation 6 is used to report the current value of a counter point. See 11.1 for a description of a Counter Point Type.

Variation 6 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

Formal structure

UINT16: Value

This is the most recently obtained or computed value.

Range is 0 to +65535.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.68 Counter—32-bit without flag, delta

DNP3 Object Library		Group:	20
		Variation:	7
Group Name:	Counter	Type:	Static
Variation Name:	32-bit without flag, delta		

Description

Object group 20, variation 7 is used to report the current value of a delta counter point.

Variation 7 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position	
								b0	Count value
b31									

Formal structure

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.69 Counter—16-bit without flag, delta

DNP3 Object Library		Group:	20
		Variation:	8
Group Name:	Counter	Type:	Static
Variation Name:	16-bit without flag, delta		

Description

Object group 20, variation 8 is used to report the current value of a delta Counter point.

Variation 8 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

Formal structure

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.70 Frozen counter—32-bit with flag

DNP3 Object Library		Group:	21
		Variation:	1
Group Name:	Frozen Counter	Type:	Static
Variation Name:	32-bit with flag		

Description

Object group 21, variation 1 is used to report the value of a counter point captured at the instant when the count is frozen. See 11.1 for a description of a Counter Point Type.

Variation 1 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	DC	RO	LF	RF	CL	RS	OL		Flag octet
									b0	
										Count value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	DISCONTINUITY
Bit 7:	Reserved, always 0.

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.71 Frozen counter—16-bit with flag

DNP3 Object Library		Group:	21
		Variation:	2
Group Name:	Frozen Counter	Type:	Static
Variation Name:	16-bit with flag		

Description

Object group 21, variation 2 is used to report the value of a counter point captured at the instant when the count is frozen. See 11.1 for a description of a Counter Point Type.

Variation 2 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	DC	RO	LF	RF	CL	RS	OL		Flag octet
									b0	Count value
b15										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	DISCONTINUITY
Bit 7:	Reserved, always 0.

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.72 Frozen counter—32-bit with flag, delta

DNP3 Object Library		Group:	21
		Variation:	3
Group Name:	Frozen Counter	Type:	Static
Variation Name:	32-bit with flag, delta		

Description

Object group 21, variation 3 is used to report the value of a delta counter point captured at the instant when the count is frozen.

Variation 3 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	RO	LF	RF	CL	RS	OL		Flag octet
									b0	
										Count value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	Reserved, always 0.
Bit 7:	Reserved, always 0.

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.73 Frozen counter—16-bit with flag, delta

DNP3 Object Library		Group:	21
		Variation:	4
Group Name:	Frozen Counter	Type:	Static
Variation Name:	16-bit with flag, delta		

Description

Object group 21, variation 4 is used to report the value of a delta counter point captured at the instant when the count is frozen.

Variation 4 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	RO	LF	RF	CL	RS	OL		Flag octet
									b0	Count value
b15										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	Reserved, always 0.
Bit 7:	Reserved, always 0.

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.74 Frozen counter—32-bit with flag and time

DNP3 Object Library		Group:	21
		Variation:	5
Group Name:	Frozen Counter	Type:	Static
Variation Name:	32-bit with flag and time		

Description

Object group 21, variation 5 is used to report the value of a counter point captured at the instant when the count is frozen. See 11.1 for a description of a Counter Point Type.

Variation 5 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	DC	RO	LF	RF	CL	RS	OL		Flag octet
									b0	
										Count value
b31										
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	DISCONTINUITY
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.75 Frozen counter—16-bit with flag and time

DNP3 Object Library		Group:	21
		Variation:	6
Group Name:	Frozen Counter	Type:	Static
Variation Name:	16-bit with flag and time		

Description

Object group 21, variation 6 is used to report the value of a counter point captured at the instant when the count is frozen. See 11.1 for a description of a Counter Point Type.

Variation 6 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	DC	RO	LF	RF	CL	RS	OL		Flag octet
									b0	Count value
b15										
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	DISCONTINUITY
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.76 Frozen counter—32-bit with flag and time, delta

DNP3 Object Library		Group:	21
		Variation:	7
Group Name:	Frozen Counter	Type:	Static
Variation Name:	32-bit with flag and time, delta		

Description

Object group 21, variation 7 is used to report the value of a delta counter point captured at the instant when the count is frozen.

Variation 7 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	RO	LF	RF	CL	RS	OL		Flag octet
									b0	
										Count value
b31										
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	Reserved, always 0.
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.77 Frozen counter—16-bit with flag and time, delta

DNP3 Object Library		Group:	21
		Variation:	8
Group Name:	Frozen Counter	Type:	Static
Variation Name:	16-bit with flag and time, delta		

Description

Object group 21, variation 8 is used to report the value of a delta counter point captured at the instant when the count is frozen.

Variation 8 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	RO	LF	RF	CL	RS	OL		Flag octet
									b0	Count value
b15										
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	Reserved, always 0.
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.78 Frozen counter—32-bit without flag

DNP3 Object Library		Group:	21
		Variation:	9
Group Name:	Frozen Counter	Type:	Static
Variation Name:	32-bit without flag		

Description

Object group 21, variation 9 is used to report the value of a counter point captured at the instant when the count is frozen. See 11.1 for a description of a Counter Point Type.

Variation 9 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

Formal structure

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.79 Frozen counter—16-bit without flag

DNP3 Object Library		Group:	21
		Variation:	10
Group Name:	Frozen Counter	Type:	Static
Variation Name:	16-bit without flag		

Description

Object group 21, variation 10 is used to report the value of a counter point captured at the instant when the count is frozen. See 11.1 for a description of a Counter Point Type.

Variation 10 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

Formal structure

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.80 Frozen counter—32-bit without flag, delta

DNP3 Object Library		Group:	21
		Variation:	11
Group Name:	Frozen Counter	Type:	Static
Variation Name:	32-bit without flag, delta		

Description

Object group 21, variation 11 is used to report the value of a delta counter point captured at the instant when the count is frozen.

Variation 11 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

Formal structure

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.81 Frozen counter—16-bit without flag, delta

DNP3 Object Library		Group:	21
		Variation:	12
Group Name:	Frozen Counter	Type:	Static
Variation Name:	16-bit without flag, delta		

Description

Object group 21, variation 12 is used to report the value of a delta counter point captured at the instant when the count is frozen.

Variation 12 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

Formal structure

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.82 Counter event—32-bit with flag

DNP3 Object Library		Group:	22
		Variation:	1
Group Name:	Counter Event	Type:	Event
Variation Name:	32-bit with flag		

Description

Object group 22, variation 1 is used to report the value of a counter point after the count has changed. See 11.1 for a description of a Counter Point Type.

Variation 1 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	DC	RO	LF	RF	CL	RS	OL		Flag octet
									b0	
										Count value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	DISCONTINUITY
Bit 7:	Reserved, always 0.

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.83 Counter event—16-bit with flag

DNP3 Object Library		Group:	22
		Variation:	2
Group Name:	Counter Event	Type:	Event
Variation Name:	16-bit with flag		

Description

Object group 22, variation 2 is used to report the value of a counter point after the count has changed. See 11.1 for a description of a Counter Point Type.

Variation 2 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	DC	RO	LF	RF	CL	RS	OL		Flag octet
									b0	Count value
b15										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	DISCONTINUITY
Bit 7:	Reserved, always 0.

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.84 Counter event—32-bit with flag, delta

DNP3 Object Library		Group:	22
		Variation:	3
Group Name:	Counter Event	Type:	Event
Variation Name:	32-bit with flag, delta		

Description

Object group 22, variation 3 is used to report the value of a delta counter point after the count has changed.

Variation 3 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	RO	LF	RF	CL	RS	OL		Flag octet
									b0	
										Count value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	Reserved, always 0.
Bit 7:	Reserved, always 0.

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.85 Counter event—16-bit with flag, delta

DNP3 Object Library		Group:	22
		Variation:	4
Group Name:	Counter Event	Type:	Event
Variation Name:	16-bit with flag, delta		

Description

Object group 22, variation 4 is used to report the value of a delta counter point after the count has changed.

Variation 4 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	RO	LF	RF	CL	RS	OL		Flag octet
									b0	Count value
b15										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	Reserved, always 0.
Bit 7:	Reserved, always 0.

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.86 Counter event—32-bit with flag and time

DNP3 Object Library		Group:	22
		Variation:	5
Group Name:	Counter Event	Type:	Event
Variation Name:	32-bit with flag and time		

Description

Object group 22, variation 5 is used to report the value of a counter point after the count has changed. See 11.1 for a description of a Counter Point Type.

Variation 5 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	DC	RO	LF	RF	CL	RS	OL		Flag octet
									b0	
										Count value
b31										
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	DISCONTINUITY
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.87 Counter event—16-bit with flag and time

DNP3 Object Library		Group:	22
		Variation:	6
Group Name:	Counter Event	Type:	Event
Variation Name:	16-bit with flag and time		

Description

Object group 22, variation 6 is used to report the value of a counter point after the count has changed. See 11.1 for a description of a Counter Point Type.

Variation 6 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	RO	LF	RF	CL	RS	OL		Flag octet
									b0	Count value
b15										
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	Reserved, always 0.
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.88 Counter event—32-bit with flag and time, delta

DNP3 Object Library		Group:	22
		Variation:	7
Group Name:	Counter Event	Type:	Event
Variation Name:	32-bit with flag and time, delta		

Description

Object group 22, variation 7 is used to report the value of a delta counter point after the count has changed.

Variation 7 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	RO	LF	RF	CL	RS	OL		Flag octet
									b0	
										Count value
b31										
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	Reserved, always 0.
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.89 Counter event—16-bit with flag and time, delta

DNP3 Object Library		Group:	22
		Variation:	8
Group Name:	Counter Event	Type:	Event
Variation Name:	16-bit with flag and time, delta		

Description

Object group 22, variation 8 is used to report the value of a delta counter point after the count has changed.

Variation 8 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	RO	LF	RF	CL	RS	OL		Flag octet
									b0	Count value
b15										
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	Reserved, always 0.
Bit 7:	Reserved, always 0.

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.90 Frozen counter event—32-bit with flag

DNP3 Object Library		Group:	23
		Variation:	1
Group Name:	Frozen Counter Event	Type:	Event
Variation Name:	32-bit with flag		

Description

Object group 23, variation 1 is used to report, as an event, the value of a counter point captured at the instant when the count is frozen. See 11.1 for a description of a Counter Point Type.

Variation 1 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	DC	RO	LF	RF	CL	RS	OL		Flag octet
									b0	
										Count value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	DISCONTINUITY
Bit 7:	Reserved, always 0.

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.91 Frozen counter event—16-bit with flag

DNP3 Object Library		Group:	23
		Variation:	2
Group Name:	Frozen Counter Event	Type:	Event
Variation Name:	16-bit with flag		

Description

Object group 23, variation 2 is used to report, as an event, the value of a counter point captured at the instant when the count is frozen. See 11.1 for a description of a Counter Point Type.

Variation 2 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	RO	LF	RF	CL	RS	OL		Flag octet
									b0	Count value
b15										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	Reserved, always 0.
Bit 7:	Reserved, always 0.

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.92 Frozen counter event—32-bit with flag, delta

DNP3 Object Library		Group:	23
		Variation:	3
Group Name:	Frozen Counter Event	Type:	Event
Variation Name:	32-bit with flag, delta		

Description

Object group 23, variation 3 is used to report, as an event, the value of a delta counter point captured at the instant when the count is frozen.

Variation 3 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	RO	LF	RF	CL	RS	OL		Flag octet
									b0	
										Count value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	Reserved, always 0.
Bit 7:	Reserved, always 0.

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.93 Frozen counter event—16-bit with flag, delta

DNP3 Object Library		Group:	23
		Variation:	4
Group Name:	Frozen Counter Event	Type:	Event
Variation Name:	16-bit with flag, delta		

Description

Object group 23, variation 4 is used to report, as an event, the value of a delta counter point captured at the instant when the count is frozen.

Variation 4 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	RO	LF	RF	CL	RS	OL		Flag octet
									b0	Count value
b15										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	Reserved, always 0.
Bit 7:	Reserved, always 0.

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.94 Frozen counter event—32-bit with flag and time

DNP3 Object Library		Group:	23
		Variation:	5
Group Name:	Frozen Counter Event	Type:	Event
Variation Name:	32-bit with flag and time		

Description

Object group 23, variation 5 is used to report, as an event, the value of a counter point captured at the instant when the count is frozen. See 11.1 for a description of a Counter Point Type.

Variation 5 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	DC	RO	LF	RF	CL	RS	OL		Flag octet
									b0	
										Count value
b31										
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	DISCONTINUITY
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.95 Frozen counter event—16-bit with flag and time

DNP3 Object Library		Group:	23
		Variation:	6
Group Name:	Frozen Counter Event	Type:	Event
Variation Name:	16-bit with flag and time		

Description

Object group 23, variation 6 is used to report, as an event, the value of a counter point captured at the instant when the count is frozen. See 11.1 for a description of a Counter Point Type.

Variation 6 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	DC	RO	LF	RF	CL	RS	OL		Flag octet
									b0	Count value
b15										
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	DISCONTINUITY
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.96 Frozen counter event—32-bit with flag and time, delta

DNP3 Object Library		Group:	23
		Variation:	7
Group Name:	Frozen Counter Event	Type:	Event
Variation Name:	32-bit with flag and time, delta		

Description

Object group 23, variation 7 is used to report, as an event, the value of a delta counter point captured at the instant when the count is frozen.

Variation 7 objects contain a 32-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	RO	LF	RF	CL	RS	OL		Flag octet
									b0	
										Count value
b31										
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	Reserved, always 0.
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT32: Count value

This is the most recently obtained or computed value.

Range is 0 to +4294967295.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.97 Frozen counter event—16-bit with flag and time, delta

DNP3 Object Library		Group:	23
		Variation:	8
Group Name:	Frozen Counter Event	Type:	Static
Variation Name:	16-bit with flag and time, delta		

Description

Object group 23, variation 8 is used to report, as an event, the value of a delta counter point captured at the instant when the count is frozen.

Variation 8 objects contain a 16-bit, unsigned integer count value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	0	RO	LF	RF	CL	RS	OL		Flag octet
									b0	Count value
b15										
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	ROLLOVER
Bit 6:	Reserved, always 0.
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT16: Count value

This is the most recently obtained or computed value.

Range is 0 to +65535.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—**Delta counters are obsolete.** New designs should not implement or specify this variation. A description is presented here for reference only.

NOTE 2—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.98 Analog input—32-bit with flag

DNP3 Object Library		Group:	30
		Variation:	1
Group Name:	Analog Input	Type:	Static
Variation Name:	32-bit with flag		

Description

Object group 30, variation 1 is used to report the current value of an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 1 objects contain a flag octet and a 32-bit, signed integer value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

INT32: Value

This is the most recently measured, obtained, or computed value.

Range is –2147483648 to +2147483647.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.99 Analog input—16-bit with flag

DNP3 Object Library		Group:	30
		Variation:	2
Group Name:	Analog Input	Type:	Static
Variation Name:	16-bit with flag		

Description

Object group 30, variation 2 is used to report the current value of an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 2 objects contain a flag octet and a 16-bit, signed integer value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	Value
b15										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

INT16: Value

This is the most recently measured, obtained, or computed value.

Range is –32768 to +32767.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.100 Analog input—32-bit without flag

DNP3 Object Library		Group:	30
		Variation:	3
Group Name:	Analog Input	Type:	Static
Variation Name:	32-bit without flag		

Description

Object group 30, variation 3 is used to report the current value of an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 3 objects contain a 32-bit, signed integer value.

Coding

Pictorial

Formal structure

INT32: Value

This is the most recently measured, obtained, or computed value.

Range is –2147483648 to +2147483647.

NOTE—This variation may be transmitted in lieu of variation 1 if the ONLINE flag would be 1 and all other flags 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.101 Analog input—16-bit without flag

DNP3 Object Library		Group:	30
		Variation:	4
Group Name:	Analog Input	Type:	Static
Variation Name:	16-bit without flag		

Description

Object group 30, variation 4 is used to report the current value of an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 4 objects contain a 16-bit, signed integer value.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position	
								b0	Value
b15									

Formal structure

INT16: Value

This is the most recently measured, obtained, or computed value.

Range is –32768 to +32767.

NOTE—This variation may be transmitted in lieu of variation 2 if the ONLINE flag would be 1 and all other flags 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.102 Analog input—Single-precision, floating-point with flag

DNP3 Object Library		Group:	30
		Variation:	5
Group Name:	Analog Input	Type:	Static
Variation Name:	Single-precision, floating-point with flag		

Description

Object group 30, variation 5 is used to report the current value of an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 5 objects contain a flag octet and a single-precision, floating-point value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

FLT32: Value

This is the most recently measured, obtained, or computed value.

Range is approximately -3.4×10^{38} to $+3.4 \times 10^{38}$.

NOTE—See 11.1 for descriptions of the flag bits and floating-point format.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.103 Analog input—Double-precision, floating-point with flag

DNP3 Object Library		Group:	30
		Variation:	6
Group Name:	Analog Input	Type:	Static
Variation Name:	Double-precision, floating-point with flag		

Description

Object group 30, variation 6 is used to report the current value of an Analog Input point. See 11.1 for a description of an Analog Input Point Type.

Variation 6 objects contain a flag octet and a double-precision, floating-point value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Value
b63										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

FLT64: Value

This is the most recently measured, obtained, or computed value.

Range is approximately -1.7×10^{308} to $+1.7 \times 10^{308}$.

NOTE—See 11.1 for descriptions of the flag bits and floating-point format.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.104 Frozen analog input—32-bit with flag

DNP3 Object Library		Group:	31
		Variation:	1
Group Name:	Frozen Analog Input	Type:	Static
Variation Name:	32-bit with flag		

Description

Object group 31, variation 1 is used to report the frozen value of an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 1 objects contain a flag octet and a 32-bit, signed integer value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Frozen value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

INT32: Frozen value

This is the analog input value at the time when it was last frozen.

Range is –2147483648 to +2147483647.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.105 Frozen analog input—16-bit with flag

DNP3 Object Library		Group:	31
		Variation:	2
Group Name:	Frozen Analog Input	Type:	Static
Variation Name:	16-bit with flag		

Description

Object group 31, variation 2 is used to report the frozen value of an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 2 objects contain a flag octet and a 16-bit, signed integer value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	Frozen value
b15										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

INT16: Frozen value

This is the analog input value at the time when it was last frozen.

Range is –32768 to +32767.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.106 Frozen analog input—32-bit with time-of-freeze

DNP3 Object Library		Group:	31
		Variation:	3
Group Name:	Frozen Analog Input	Type:	Static
Variation Name:	32-bit with time-of-freeze		

Description

Object group 31, variation 3 is used to report the frozen value of an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 3 objects contain a flag octet, a 32-bit, signed integer value, and a time-of-freeze.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Frozen value
b31										
									b0	
										Time-of-freeze
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

INT32: Frozen value

This is the analog input value at the time when it was last frozen.

Range is –2147483648 to +2147483647.

UINT48: Time-of-freeze

Time when the freeze occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.107 Frozen analog input—16-bit with time-of-freeze

DNP3 Object Library		Group:	31
		Variation:	4
Group Name:	Frozen Analog Input	Type:	Static
Variation Name:	16-bit with time-of-freeze		

Description

Object group 31, variation 4 is used to report the frozen value of an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 4 objects contain a flag octet, a 16-bit, signed integer value, and a time-of-freeze.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	Frozen value
b15										
									b0	
										Time-of-freeze
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

INT16: Frozen value

This is the analog input value at the time when it was last frozen.

Range is –32768 to +32767.

UINT48: Time-of-occurrence

Time when the freeze occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.108 Frozen analog input—32-bit without flag

DNP3 Object Library		Group:	31
		Variation:	5
Group Name:	Frozen Analog Input	Type:	Static
Variation Name:	32-bit without flag		

Description

Object group 31, variation 5 is used to report the frozen value of an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 5 objects contain a 32-bit, signed integer value.

Coding

Pictorial

Formal structure

INT32: Frozen value

This is the analog input value at the time when it was last frozen.

Range is –2147483648 to +2147483647.

NOTE—This variation may be transmitted in lieu of variation 1 if the ONLINE flag would be 1 and all other flags 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.109 Frozen analog input—16-bit without flag

DNP3 Object Library		Group:	31
		Variation:	6
Group Name:	Frozen Analog Input	Type:	Static
Variation Name:	16-bit without flag		

Description

Object group 31, variation 6 is used to report the frozen value of an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 6 objects contain a 16-bit, signed integer value.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
								b0
b15								

Frozen value

Formal structure

INT16: Frozen value

This is the analog input value at the time when it was last frozen.

Range is –32768 to +32767.

NOTE—This variation may be transmitted in lieu of variation 2 if the ONLINE flag would be 1 and all other flags 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.110 Frozen analog input—Single-precision, floating-point with flag

DNP3 Object Library		Group:	31
		Variation:	7
Group Name:	Frozen Analog Input	Type:	Static
Variation Name:	Single-precision, floating-point with flag		

Description

Object group 31, variation 7 is used to report the frozen value of an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 7 objects contain a flag octet and a single-precision, floating-point value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Frozen value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

FLT32: Frozen value

This is the analog input value at the time when it was last frozen.

Range is approximately -3.4×10^{38} to $+3.4 \times 10^{38}$.

NOTE—See 11.1 for descriptions of the flag bits and floating-point format.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.111 Frozen analog input—Double-precision, floating-point with flag

DNP3 Object Library		Group:	31
		Variation:	8
Group Name:	Frozen Analog Input	Type:	Static
Variation Name:	Double-precision, floating-point with flag		

Description

Object group 31, variation 8 is used to report the frozen value of an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 8 objects contain a flag octet and a double-precision, floating-point value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Frozen value
b63										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

FLT64: Frozen value

This is the analog input value at the time when it was last frozen.

Range is approximately -1.7×10^{308} to $+1.7 \times 10^{308}$.

NOTE—See 11.1 for descriptions of the flag bits and floating-point format.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.112 Analog input event—32-bit without time

DNP3 Object Library		Group:	32
		Variation:	1
Group Name:	Analog Input Event	Type:	Event
Variation Name:	32-bit without time		

Description

Object group 32, variation 1 is used to report events related to an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 1 objects contain a flag octet and a 32-bit, signed integer value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

INT32: Value

This is the most recently measured, obtained, or computed value.

Range is –2147483648 to +2147483647.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.113 Analog input event—16-bit without time

DNP3 Object Library		Group:	32
		Variation:	2
Group Name:	Analog Input Event	Type:	Event
Variation Name:	16-bit without time		

Description

Object group 32, variation 2 is used to report events related to an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 2 objects contain a flag octet and a 16-bit, signed integer value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	Value
b15										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

INT16: Value

This is the most recently measured, obtained, or computed value.

Range is –32768 to +32767.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.114 Analog input event—32-bit with time

DNP3 Object Library		Group:	32
		Variation:	3
Group Name:	Analog Input Event	Type:	Event
Variation Name:	32-bit with time		

Description

Object group 32, variation 3 is used to report events related to an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 3 objects contain a flag octet, a 32-bit, signed integer value, and a time-of-occurrence.

Coding

Pictorial

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

INT32: Value

This is the most recently measured, obtained, or computed value.

Range is –2147483648 to +2147483647.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.115 Analog input event—16-bit with time

DNP3 Object Library		Group:	32
		Variation:	4
Group Name:	Analog Input Event	Type:	Event
Variation Name:	16-bit with time		

Description

Object group 32, variation 4 is used to report events related to an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 4 objects contain a flag octet, a 16-bit, signed integer value, and a time-of-occurrence.

Coding

Pictorial

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

INT16: Value

This is the most recently measured, obtained, or computed value.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Range is –32768 to +32767.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.116 Analog input event—Single-precision, floating-point without time

DNP3 Object Library		Group:	32
		Variation:	5
Group Name:	Analog Input Event	Type:	Event
Variation Name:	Single-precision, floating-point without time		

Description

Object group 32, variation 5 is used to report events related to an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 5 objects contain a flag octet and a single-precision, floating-point value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

FLT32: Value

This is the most recently measured, obtained, or computed value.

Range is approximately -3.4×10^{38} to $+3.4 \times 10^{38}$.

NOTE—See 11.1 for descriptions of the flag bits and floating-point format.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.117 Analog input event—Double-precision, floating-point without time

DNP3 Object Library		Group:	32
		Variation:	6
Group Name:	Analog Input Event	Type:	Event
Variation Name:	Double-precision, floating-point without time		

Description

Object group 32, variation 6 is used to report change events related to an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 6 objects contain a flag octet and a double-precision, floating-point value.

Coding

Pictorial

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

FLT64: Value

This is the most recently measured, obtained, or computed value.

Range is approximately -1.7×10^{308} to $+1.7 \times 10^{308}$.

NOTE—See 11.1 for descriptions of the flag bits and floating-point format.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.118 Analog input event—Single-precision, floating-point with time

DNP3 Object Library		Group:	32
		Variation:	7
Group Name:	Analog Input Event	Type:	Event
Variation Name:	Single-precision, floating-point with time		

Description

Object group 32, variation 7 is used to report events related to an analog input point. See 11.1 for a description of a Analog Input Point Type.

Variation 7 objects contain a flag octet, a single-precision, floating-point value, and a time-of-occurrence.

Coding

Pictorial

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

FLT32: Value

This is the most recently measured, obtained, or computed value.

Range is approximately -3.4×10^{38} to $+3.4 \times 10^{38}$.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for descriptions of the flag bits and floating-point format.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.119 Analog input event—Double-precision, floating-point with time

DNP3 Object Library		Group:	32
		Variation:	8
Group Name:	Analog Input Event	Type:	Event
Variation Name:	Double-precision, floating-point with time		

Description

Object group 32, variation 8 is used to report events related to an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 8 objects contain a flag octet, a double-precision, floating-point value, and a time-of-occurrence.

Coding

Pictorial

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Bit 7: Reserved, always 0.

FLT32: Value

This is the most recently measured, obtained, or computed value.

Range is approximately -1.7×10^{308} to $+1.7 \times 10^{308}$.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for descriptions of the flag bits and floating-point format.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.120 Frozen analog input event—32-bit without time

DNP3 Object Library		Group:	33
		Variation:	1
Group Name:	Frozen Analog Input Event	Type:	Event
Variation Name:	32-bit without time		

Description

Object group 33, variation 1 is used to report frozen value change events related to an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 1 objects contain a flag octet and a 32-bit, signed integer value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	Frozen value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

INT32: Frozen value

This is the analog input value at the time when it was last frozen.

Range is –2147483648 to +2147483647.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.121 Frozen analog input event—16-bit without time

DNP3 Object Library		Group:	33
		Variation:	2
Group Name:	Frozen Analog Input Event	Type:	Event
Variation Name:	16-bit without time		

Description

Object group 33, variation 2 is used to report frozen value change events related to an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 2 objects contain a flag octet and a 16-bit, signed integer value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	Frozen value
b15										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

INT16: Frozen value

This is the analog input value at the time when it was last frozen.

Range is –32768 to +32767.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.122 Frozen analog input event—32-bit with time

DNP3 Object Library		Group:	33
		Variation:	3
Group Name:	Frozen Analog Input Event	Type:	Event
Variation Name:	32-bit with time		

Description

Object group 33, variation 3 is used to report frozen value change events related to an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 3 objects contain a flag octet, a 32-bit, signed integer value, and a time-of-freeze.

Coding

Pictorial

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

INT32: Frozen value

This is the analog input value at the time when it was last frozen.

Range is –2147483648 to +2147483647.

UINT48: Time-of-freeze

Time when the freeze occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.123 Frozen analog input event—16-bit with time

DNP3 Object Library		Group:	33
		Variation:	4
Group Name:	Frozen Analog Input Event	Type:	Event
Variation Name:	16-bit with time		

Description

Object group 33, variation 4 is used to report frozen value change events related to an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 4 objects contain a flag octet, a 16-bit, signed integer value, and a time-of-freeze.

Coding

Pictorial

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

INT16: Frozen value

This is the analog input value at the time when it was last frozen.

Range is –32768 to +32767.

UINT48: Time-of-freeze

Time when the freeze occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for flag bit descriptions.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.124 Frozen analog input event—Single-precision, floating-point without time

DNP3 Object Library		Group:	33
		Variation:	5
Group Name:	Frozen Analog Input Event	Type:	Event
Variation Name:	Single-precision, floating-point without time		

Description

Object group 33, variation 5 is used to report frozen value change events related to an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 5 objects contain a flag octet and a single-precision, floating-point value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Frozen value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

FLT32: Frozen value

This is the analog input value at the time when it was last frozen.

Range is approximately -3.4×10^{38} to $+3.4 \times 10^{38}$.

NOTE—See 11.1 for descriptions of the flag bits and floating-point format.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.125 Frozen analog input event—Double-precision, floating-point without time

DNP3 Object Library		Group:	33
		Variation:	6
Group Name:	Frozen Analog Input Change Event	Type:	Event
Variation Name:	Double-precision, floating-point without time		

Description

Object group 33, variation 6 is used to report frozen value change events related to an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 6 objects contain a flag octet and a double-precision, floating-point value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Frozen value
b63										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

FLT64: Frozen value

This is the analog input value at the time when it was last frozen.

Range is approximately -1.7×10^{308} to $+1.7 \times 10^{308}$.

NOTE—See 11.1 for descriptions of the flag bits and floating-point format.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.126 Frozen analog input event—Single-precision, floating-point with time

DNP3 Object Library		Group:	33
		Variation:	7
Group Name:	Frozen Analog Input Event	Type:	Event
Variation Name:	Single-precision, floating-point with time		

Description

Object group 33, variation 7 is used to report frozen value change events related to an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 7 objects contain a flag octet, a single-precision, floating-point value, and a time-of-freeze.

Coding

Pictorial

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

FLT32: Frozen value

This is the analog input value at the time when it was last frozen.

Range is approximately -3.4×10^{38} to $+3.4 \times 10^{38}$.

UINT48: Time-of-freeze

Time when the freeze occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for descriptions of the flag bits and floating-point format.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.127 Frozen analog input event—Double-precision, floating-point with time

DNP3 Object Library		Group:	33
		Variation:	8
Group Name:	Frozen Analog Input Event	Type:	Event
Variation Name:	Double-precision, floating-point with time		

Description

Object group 33, variation 8 is used to report frozen value change events related to an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 8 objects contain a flag octet, a double-precision, floating-point value, and a time-of-freeze.

Coding

Pictorial

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Bit 7: Reserved, always 0.

FLT32: Frozen value

This is the analog input value at the time when it was last frozen.

Range is approximately -1.7×10^{308} to $+1.7 \times 10^{308}$.

UINT48: Time-of-occurrence

Time when the freeze occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE—See 11.1 for descriptions of the flag bits and floating-point format.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.128 Analog input reporting deadband—16-bit

DNP3 Object Library		Group:	34
		Variation:	1
Group Name:	Analog Input Reporting Deadband	Type:	Static
Variation Name:	16-bit		

Description

Object group 34, variation 1 is used to set and report the deadband value of an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 1 objects contain a 16-bit, unsigned integer value.

Analog input reporting deadbands are readable and writeable.

A deadband of zero permits any change in the analog input value to generate an event, and a deadband of the full range of the variable prevents generation of an event.

Coding

Pictorial

Formal structure

UINT16: Deadband value

This is the deadband value expressed as

— Counts if the fixed deadbanding method is employed.

— Count-seconds if the integrating deadbanding method is employed.

Range is 0 to +65535.

NOTE 1—Objects of this type are not to be reported in read requests for class 0 data. They may only be returned in a response to a read request that specifically asks for object group 34.

NOTE 2—An outstation device should report the value of 65 535 if the deadband value is greater than or equal to 65 535 when a read request for this variation is received.

NOTE 3—The response to a read request for object group 34, variation 0 may include any variation of object group 34, in accordance with the device's capabilities and configuration.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.129 Analog input reporting deadband—32-bit

DNP3 Object Library		Group:	34
		Variation:	2
Group Name:	Analog Input Reporting Deadband	Type:	Static
Variation Name:	32-bit		

Description

Object group 34, variation 2 is used to set and report the deadband value of an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 2 objects contain a 32-bit, unsigned integer value.

Analog input reporting deadbands are readable and writeable.

A deadband of zero permits any change in the analog input value to generate an event, and a deadband of the full range of the variable prevents generation of an event.

Coding

Pictorial

Formal structure

UINT32: Deadband value

This is the deadband value expressed as

— Counts if the fixed deadbanding method is employed.

— Count-seconds if the integrating deadbanding method is employed.

Range is 0 to +4294967295.

NOTE 1—Objects of this type are not to be reported in read requests for class 0 data. They may only be returned in a response to a read request that specifically asks for object group 34.

NOTE 2—An outstation device should report the value of 4 294 967 295 if the deadband value is greater than or equal to 4 294 967 295 when a read request for this variation is received.

NOTE 3—The response to a read request for object group 34, variation 0 may include any variation of object group 34, in accordance with the device's capabilities and configuration.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.130 Analog input reporting deadband—Single-precision, floating-point

DNP3 Object Library		Group:	34
		Variation:	3
Group Name:	Analog Input Reporting Deadband	Type:	Static
Variation Name:	Single-precision, floating-point		

Description

Object group 34, variation 3 is used to set and report the deadband value of an analog input point. See 11.1 for a description of an Analog Input Point Type.

Variation 3 objects contain a positive single-precision, floating-point value.

Analog input reporting deadbands are readable and writeable.

A deadband of zero permits any change in the analog input value to generate an event, and a deadband of the full range of the variable prevents generation of an event.

Coding

Pictorial

Formal structure

FLT32: Deadband value

This is the deadband value expressed as

— Engineering units if the fixed deadbanding method is employed.

— Engineering unit-seconds if the integrating deadbanding method is employed.

Range is approximately 0 to $+3.4 \times 10^{38}$. Note that even though floating-point numbers can have negative values, only positives are permitted in this variation.

NOTE 1—Objects of this type are not to be reported in read requests for class 0 data. They may only be returned in a response to a read request that specifically asks for object group 34.

NOTE 2—A deadband value of positive infinity prevents event generation. A negative floating-point deadband value is equivalent to a zero-value deadband, as any absolute value change in the input is larger than the deadband.

NOTE 3—The response to a read request for object group 34, variation 0 may include any variation of object group 34, in accordance with the device's capabilities and configuration.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

NOTE 4—Note that if the device is capable of reporting variation 3, floating-point, the device should be configurable to only report data using variations 1 and 2 for masters that do not support floating-point.

NOTE 5—See 11.1 for a description of floating-point format.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.131 Analog output status—32-bit with flag

DNP3 Object Library		Group:	40
		Variation:	1
Group Name:	Analog Output Status	Type:	Static
Variation Name:	32-bit with flag		

Description

Object group 40, variation 1 is used to report the status of an analog output point. See 11.1 for a description of an Analog Output Point Type.

Variation 1 objects contain a flag octet and a 32-bit, signed integer value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

INT32: Value

This is the analog value currently being output.

Range is –2147483648 to +2147483647.

NOTE 1—See 11.1 for flag bit descriptions.

NOTE 2—A point index in object group 40 corresponds to the same physical or logical point as the identical index in object groups 41, 42, and 43.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.132 Analog output status—16-bit with flag

DNP3 Object Library		Group:	40
		Variation:	2
Group Name:	Analog Output Status	Type:	Static
Variation Name:	16-bit with flag		

Description

Object group 40, variation 2 is used to report the status of an analog output point. See 11.1 for a description of an Analog Output Point Type.

Variation 2 objects contain a flag octet and a 16-bit, signed integer value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	Value
b15										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

INT16: Value

This is the analog value currently being output.

Range is –32768 to +32767.

NOTE 1—See 11.1 for flag bit descriptions.

NOTE 2—A point index in object group 40 corresponds to the same physical or logical point as the identical index in object groups 41, 42, and 43.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.133 Analog output status—Single-precision, floating-point with flag

DNP3 Object Library		Group:	40
		Variation:	3
Group Name:	Analog Output Status	Type:	Static
Variation Name:	Single-precision, floating-point with flag		

Description

Object group 40, variation 3 is used to report the status of an analog output point. See 11.1 for a description of an Analog Output Point Type.

Variation 3 objects contain a flag octet and a single-precision, floating-point value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

FLT32: Value

This is the analog value currently being output.

Range is approximately -3.4×10^{38} to $+3.4 \times 10^{38}$.

NOTE 1—See 11.1 for descriptions of the flag bits and floating-point format.

NOTE 2—A point index in object group 40 corresponds to the same physical or logical point as the identical index in object groups 41, 42, and 43.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.134 Analog output status—Double-precision, floating-point with flag

DNP3 Object Library		Group:	40
		Variation:	4
Group Name:	Analog Output Status	Type:	Static
Variation Name:	Double-precision, floating-point with flag		

Description

Object group 40, variation 4 is used to report the status of an analog output point. See 11.1 for a description of an Analog Output Point Type.

Variation 4 objects contain a flag octet and a double-precision, floating-point value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Value
b63										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

FLT64: Value

This is the analog value currently being output.

Range is approximately -1.7×10^{308} to $+1.7 \times 10^{308}$.

NOTE 1—See 11.1 for descriptions of the flag bits and floating-point format.

NOTE 2—A point index in object group 40 corresponds to the same physical or logical point as the identical index in object groups 41, 42, and 43.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.135 Analog output—32-bit

DNP3 Object Library		Group:	41
		Variation:	1
Group Name:	Analog Output	Type:	Cmnd
Variation Name:	32-bit		

Description

Object group 41, variation 1 is used to set an analog value into an analog output point. See 11.1 for a description of an Analog Output Point Type.

Executing controls are initiated by sending one or more request messages from a master with an appropriate Application Layer function code, ***select*** (3), ***operate*** (4), ***direct operate*** (5), or ***direct operate without acknowledgment*** (6), along with a g41v1 object for each point. This object may not be used with a ***write*** function code, 2.

Variation 1 objects contain a 32-bit, signed integer value and a control status octet.

Coding

Pictorial

Formal structure

INT32: Requested value

This is the analog value that is requested; it may be scaled and/or manipulated before a physical or pseudo analog output is set.

Range is –2147483648 to +2147483647.

UINT8: Control status

This value is always 0 in a request message.

In response messages, this value represents the status of the requested control operation. Status codes are enumerated in 11.1.

Range is 0 to 255.

NOTE—A point index in object group 41 corresponds to the same physical or logical point as the identical index in object groups 40, 42, and 43.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.136 Analog output—16-bit

DNP3 Object Library		Group:	41
		Variation:	2
Group Name:	Analog Output	Type:	Cmnd
Variation Name:	16-bit		

Description

Object group 41, variation 2 is used to set an analog value into an analog output point. See 11.1 for a description of an Analog Output Point Type.

Executing controls are initiated by sending one or more request messages from a master with an appropriate Application Layer function code, ***select*** (3), ***operate*** (4), ***direct operate*** (5), or ***direct operate without acknowledgment*** (6), along with a g41v2 object for each point. This object may not be used with a ***write*** function code, 2.

Variation 2 objects contain a 16-bit, signed integer value and a control status octet.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
									b0	Requested value
b15										
b7									b0	Control status

Formal structure

INT16: Requested value

This is the analog value that is requested; it may be scaled and/or manipulated before a physical or pseudo analog output is set.

Range is –32768 to +32767.

UINT8: Control status

This value is always 0 in a request message.

In response messages, this value represents the status of the requested control operation. Status codes are enumerated in 11.1.

Range is 0 to 255.

NOTE—A point index in object group 41 corresponds to the same physical or logical point as the identical index in object groups 40, 42, and 43.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.137 Analog output—Single-precision, floating-point

DNP3 Object Library		Group:	41
		Variation:	3
Group Name:	Analog Output	Type:	Cmnd
Variation Name:	Single-precision, floating-point		

Description

Object group 41, variation 3 is used to set an analog value into an analog output point. See 11.1 for a description of an Analog Output Point Type.

Executing controls are initiated by sending one or more request messages from a master with an appropriate Application Layer function code, ***select*** (3), ***operate*** (4), ***direct operate*** (5), or ***direct operate without acknowledgment*** (6), along with a g41v3 object for each point. This object may not be used with a ***write*** function code, 2.

Variation 3 objects contain a single-precision, floating-point value and a control status octet.

Coding

Pictorial

Formal structure

FLT32: Requested value

This is the analog value that is requested; it may be scaled and/or manipulated before a physical or pseudo analog output is set.

Range is approximately -3.4×10^{38} to $+3.4 \times 10^{38}$.

UINT8: Control status

This value is always 0 in a request message.

In response messages, this value represents the status of the requested control operation. Status codes are enumerated in 11.1.

Range is 0 to 255.

NOTE 1—See 11.1 for a description of the floating-point format.

NOTE 2—A point index in object group 41 corresponds to the same physical or logical point as the identical index in object groups 40, 42, and 43.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.138 Analog output—Double-precision, floating-point

DNP3 Object Library		Group:	41
		Variation:	4
Group Name:	Analog Output	Type:	Cmnd
Variation Name:	Double-precision, floating-point		

Description

Object group 41, variation 4 is used to set an analog value into an analog output point. See 11.1 for a description of an Analog Output Point Type.

Executing controls are initiated by sending one or more request messages from a master with an appropriate Application Layer function code, ***select*** (3), ***operate*** (4), ***direct operate*** (5), or ***direct operate without acknowledgment*** (6), along with a g41v4 object for each point. This object may not be used with a ***write*** function code, 2.

Variation 4 objects contain a double-precision, floating-point value and a control status octet.

Coding

Pictorial

Formal structure

FLT32: Requested value

This is the analog value that is requested; it may be scaled and/or manipulated before a physical or pseudo analog output is set.

Range is approximately -1.7×10^{308} to $+1.7 \times 10^{308}$.

UINT8: Control status

This value is always 0 in a request message.

In response messages, this value represents the status of the requested control operation. Status codes are enumerated in 11.1.

Range is 0 to 255.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

NOTE 1—See 11.1 for a description of the floating-point format.

NOTE 2—A point index in object group 41 corresponds to the same physical or logical point as the identical index in object groups 40, 42, and 43.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.139 Analog output event—32-bit without time

DNP3 Object Library		Group:	42
		Variation:	1
Group Name:	Analog Output Event	Type:	Event
Variation Name:	32-bit without time		

Description

An Analog Output Event Object is an instance of a report for an outstation's corresponding Analog Output Status object (object group 40). See 11.1 for a description of an Analog Output Point Type. An Analog Output Event may be generated for a point only if the outstation returns an output status object for that point in response to a static (class 0) request. A event may be generated by an outstation when it detects something of interest has occurred. Examples of this include a change in output point value or a change in output object flags. An analog output event object may also be generated due to other application-dependent reasons.

For analog output points, changes to the exception flags, and changes of the output value when the underlying point retains a value, are reported with Analog Output Event objects. The analog output point's value and status at the time when the event is generated and queued are the value and status placed into the Analog Output Event object.

This object shall not be generated to simply report that a command was received. Object Group 43 objects should be used for this purpose.

Generation of Analog Output Events is optional for an outstation and may occur in one or more of the following situations:

— Output point value changed—upon a control from a master or upstream device that results in a change to the value or status of an output point, the outstation may generate an event object for that point. For example, where multiple masters control the same output point on an outstation, a change event may be generated to one or more master stations to indicate that the output changed value.

— External effects—mechanisms other than a control from a master may result in a change to the value of a point represented by an output object. An event may be generated to indicate to the master that the output point value has changed. For example, a local automation sequence may override the value of a point set by a previous control from the master.

— Flags changed—where an outstation generates Analog Output Events for a particular point, an event shall be generated when any of the flags for the output point change. Note that the Flags octet relates to the output point status and not to the status value of any related input point(s).

Outstations that implement this object:

— Shall be able to configure whether output change events are generated on output points.

— Should support Application Layer Function Code 22 (ASSIGN_CLASS) for Object Group 40, Variation 0 if the outstation device supports Assign Class functions on other objects.

When events are configured for a particular output point, the outstation:

— May generate an event when something of interest happens on the output point.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— Shall generate an event when an output point's output value changes significantly.

— When flags are supported for the point, shall generate an event when any of the status flags for the output point change.

Variation 1 objects contain a flag octet that indicates the status of the output point and a 32-bit signed integer value.

Coding

Pictorial

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0

INT32: Value

This is the most recently measured, obtained, or computed output value.

Range is –2147483648 to +2147483647.

NOTE 1—A point index in object group 42 corresponds to the same physical or logical point as the identical index in object groups 40, 41, and 43.

NOTE 2—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 40 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.140 Analog output event—16-bit without time

DNP3 Object Library		Group:	42
		Variation:	2
Group Name:	Analog Output Event	Type:	Event
Variation Name:	16-bit without time		

Description

See object group 42, variation 1 for a description of Analog Output Events and event detection.

Variation 2 objects contain a flag octet and a 16-bit signed integer value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	Value
b15										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0

INT16: Value

This is the most recently measured, obtained, or computed output value.

Range is –32768 to +32767.

NOTE 1—A point index in object group 42 corresponds to the same physical or logical point as the identical index in object groups 40, 41, and 43.

NOTE 2—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 40 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.141 Analog output event—32-bit with time

DNP3 Object Library		Group:	42
		Variation:	3
Group Name:	Analog Output Event	Type:	Event
Variation Name:	32-bit with time		

Description

See object group 42, variation 1 for a description of Analog Output Events and event detection.

Variation 3 objects contain a flag octet, a 32-bit signed integer value, and a time-of-occurrence.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Value
b31										
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

INT32: Value

This is the most recently measured, obtained, or computed output value.

Range is –2147483648 to +2147483647.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—A point index in object group 42 corresponds to the same physical or logical point as the identical index in object groups 40, 41, and 43.

NOTE 2—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 40 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.142 Analog output event—16-bit with time

DNP3 Object Library		Group:	42
		Variation:	4
Group Name:	Analog Output Event	Type:	Event
Variation Name:	16-bit with time		

Description

See object group 42, variation 1 for a description of Analog Output Events and event detection.

Variation 4 objects contain a flag octet, a 16-bit signed integer value, and a time-of-occurrence.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	Value
b15										
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0

INT16: Value

This is the most recently measured, obtained, or computed output value.

Range is –32768 to +32767.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—A point index in object group 42 corresponds to the same physical or logical point as the identical index in object groups 40, 41, and 43.

NOTE 2—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 40 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.143 Analog output event—Single-precision, floating-point without time

DNP3 Object Library		Group:	42
		Variation:	5
Group Name:	Analog Output Event	Type:	Event
Variation Name:	Single-precision, floating-point without time		

Description

See object group 42, variation 1 for a description of Analog Output Events and event detection.

Variation 5 objects contain a flag octet and a single-precision floating-point value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Value
b31										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERERNCE_ERR
Bit 7:	Reserved, always 0

FLT32: Value

This is the most recently measured, obtained, or computed output value.

Range is approximately -3.4×10^{38} to $+3.4 \times 10^{38}$.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

NOTE 1—See 11.1 for a description of the floating-point format.

NOTE 2—A point index in object group 42 corresponds to the same physical or logical point as the identical index in object groups 40, 41, and 43.

NOTE 3—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 40 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.144 Analog output event—Double-precision, floating-point without time

DNP3 Object Library		Group:	42
		Variation:	6
Group Name:	Analog Output Event	Type:	Event
Variation Name:	Double-precision, floating-point without time		

Description

See object group 42, variation 1 for a description of Analog Output Events and event detection.

Variation 6 objects contain a flag octet and a single-precision floating-point value.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Value
b63										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0

FLT64: Value

This is the most recently measured, obtained, or computed output value.

Range is approximately -1.7×10^{308} to $+1.7 \times 10^{308}$.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

NOTE 1—See 11.1 for a description of the floating-point format.

NOTE 2—A point index in object group 42 corresponds to the same physical or logical point as the identical index in object groups 40, 41, and 43.

NOTE 3—This object group's events are assigned to a specific event class using an assign class message with objects from object group 40 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.145 Analog output event—Single-precision, floating-point with time

DNP3 Object Library		Group:	42
		Variation:	7
Group Name:	Analog Output Event	Type:	Event
Variation Name:	Single-precision, floating-point with time		

Description

See object group 42, variation 1 for a description of Analog Output Events and event detection.

Variation 7 objects contain a flag octet, a single-precision floating-point value, and a time-of-occurrence.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Value
b31										
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

FLT32: Value

This is the most recently measured, obtained, or computed output value.

Range is approximately -3.4×10^{38} to $+3.4 \times 10^{38}$.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—See 11.1 for a description of the floating-point format.

NOTE 2—A point index in object group 42 corresponds to the same physical or logical point as the identical index in object groups 40, 41, and 43.

NOTE 3—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 40 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.146 Analog output event—Double-precision, floating-point with time

DNP3 Object Library		Group:	42
		Variation:	8
Group Name:	Analog Output Event	Type:	Event
Variation Name:	Double-precision, floating-point with time		

Description

See object group 42, variation 1 for a description of Analog Output Events and event detection.

Variation 8 objects contain a flag octet, a double-precision floating-point value, and a time-of-occurrence.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	RE	OR	LF	RF	CL	RS	OL		Flag octet
									b0	
										Value
b63										
									b0	
										Time-of-occurrence
b47										

Formal structure

BSTR8: Flag Octet

Bit 0:	ONLINE
Bit 1:	RESTART
Bit 2:	COMM_LOST
Bit 3:	REMOTE_FORCED
Bit 4:	LOCAL_FORCED
Bit 5:	OVER_RANGE
Bit 6:	REFERENCE_ERR
Bit 7:	Reserved, always 0

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

FLT64: Value

This is the most recently measured, obtained, or computed output value.

Range is approximately -1.7×10^{308} to $+1.7 \times 10^{308}$.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—See 11.1 for a description of the floating-point format.

NOTE 2—A point index in object group 42 corresponds to the same physical or logical point as the identical index in object groups 40, 41, and 43.

NOTE 3—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 40 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.147 Analog output command event—32-bit without time

DNP3 Object Library		Group:	43
		Variation:	1
Group Name:	Analog Output Command Event	Type:	Event
Variation Name:	32-bit without time		

Description

An Analog Output Command Event object reports that a command has been attempted on an outstation's corresponding Analog Output point. Examples of usage of this object include:

— Reporting to a master device that an outstation has received a control from another master

— Reporting to a master device that a successful or unsuccessful control was attempted

— Reporting to a master device that a control request was made from an internal application

— Reporting to a master device that a control was received and processed at an outstation when issued through a non-originating device (e.g., Data Concentrator)

A command event may be generated by an outstation when it receives a control command for an output point from an internal or external source. Examples of this include a command request received though a DNP3 Analog Output Block, a command request from a different protocol, or a control change command from an executing application.

Where the analog output point controlled retains an output value, a control request to the output point would include value information. This requested control value is returned in this analog output command event. This object does not return value or status information for the output point itself, or any feedback input associated with the control. Output command event objects received by a master are intended to be used for informational purposes, such as being logged.

Generation of Analog Output Command Events is optional for an outstation. This object may be used in conjunction with Analog Output Change Event objects.

Outstations that implement this object:

— Shall be able to configure whether output command events are generated on output points.

— Should support Application Layer Function Code 22 (ASSIGN_CLASS) for object group 41, Variation 0 if the outstation device supports Assign Class functions on other objects.

— May generate events when controls are commanded from internal sources (e.g., application), external sources requesting controls (e.g., protocol control request), or both.

— Shall not generate this event until a command is fully processed. For example when receiving a Select Before Operate control request, an event shall not be generated following reception of a Select command until either the Select fails or the Operate is processed.

— Shall not generate this event as a result of output point value change or status change. Object group 42 objects are used for this purpose. Object group 43 objects are independent from object group 42 objects.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

— Should provide the result of processing the control request in the control status field of this object. Where this status can be provided, it shall be in the same format as the control status field provided in DNP3 object 41, variation 1.

— Should provide at least an indication of whether the processed control request was accepted (status = 0) or not accepted for unknown reasons (status = 127). Where possible, it is recommended that the outstation return known, relevant status codes. Where the processed control was received through a DNP3 request, for example, the status code returned in the output command event status code should be the same the status code used in the response to the control request.

— May choose to send output command events (1) only when the control actually succeeds (i.e., status code = 0), or (2) for any control command regardless of whether it succeeds or fails (i.e., status code >= 0).

Variation 1 objects contain value information regarding the commanded value of the output (where the point retains value information) and status information indicating the status that resulted when the outstation processed the requested control on the point. Information provided in this object relates to a control request, and not to the resultant value or status of the point controlled.

Coding

Pictorial

Formal structure

UINT7: Status Code

This value represents the status from processing the command that this event object is reporting. An enumeration of status codes appears in 11.1.

BSTR1: Reserved

Always 0

INT32: Commanded Value

The Commanded Value representing the control requested for the physical or logical output.

Range is –2147483648 to +2147483647. Where the commanded value is unknown, this field shall contain 0.

NOTE 1—A point index in object group 43 corresponds to the same physical or logical point as the identical index in object groups 40, 41, and 42.

NOTE 2—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 41 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.148 Analog output command event—16-bit without time

DNP3 Object Library		Group:	43
		Variation:	2
Group Name:	Analog Output Command Event	Type:	Event
Variation Name:	16-bit without time		

Description

See object group 43, variation 1 for a description of Analog Output Command Events and event detection.

Variation 2 objects contain a status octet that resulted when the outstation processed the requested control, and the commanded value. Information provided in this object relates to a control request, and not to the resultant state or status of the point controlled.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	0	Status code								Status
									b0	Commanded Value
b15										

Formal structure

UINT7: Status Code

This value represents the status from processing the command that this event object is reporting. An enumeration of status codes appears in 11.1.

BSTR1: Reserved

Always 0

INT16: Commanded Value

The Commanded Value representing the control requested for the physical or logical output.

Range is –32768 to +32767. Where the commanded value is unknown, this field shall contain 0.

NOTE 1—A point index in object group 43 corresponds to the same physical or logical point as the identical index in object groups 40, 41, and 42.

NOTE 2—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 41 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.149 Analog output command event—32-bit with time

DNP3 Object Library		Group:	43
		Variation:	3
Group Name:	Analog Output Command Event	Type:	Event
Variation Name:	32-bit with time		

Description

See object group 43, variation 1 for a description of Analog Output Command Events and event detection.

Variation 3 objects contain a status octet that resulted when the outstation processed the requested control, the commanded value, and a time-of-occurrence. The time-of-occurrence represents the time at which the control request was processed. Information provided in this object relates to a control request, and not to the resultant state or status of the point controlled.

Coding

Pictorial

Formal structure

UINT7: Status Code

This value represents the status from processing the command that this event object is reporting. An enumeration of status codes appears in 11.1.

BSTR1: Reserved

Always 0

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

INT32: Commanded Value

The Commanded Value representing the control requested for the physical or logical output.

Range is –2147483648 to +2147483647. Where the commanded value is unknown, this field shall contain 0.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—A point index in object group 43 corresponds to the same physical or logical point as the identical index in object groups 40, 41, and 42.

NOTE 2—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 41 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.150 Analog output command event—16-bit with time

DNP3 Object Library		Group:	43
		Variation:	4
Group Name:	Analog Output Command Event	Type:	Event
Variation Name:	16-bit with time		

Description

See object group 43, variation 1 for a description of Analog Output Command Events and event detection.

Variation 4 objects contain a status octet that resulted when the outstation processed the requested control, the commanded value, and a time-of-occurrence. The time-of-occurrence represents the time at which the control request was processed. Information provided in this object relates to a control request, and not to the resultant state or status of the point controlled.

Coding

Pictorial

Formal structure

UINT7: Status Code

This value represents the status from processing the command that this event object is reporting. An enumeration of status codes appears in 11.1.

BSTR1: Reserved

Always 0

INT16: Commanded Value

The Commanded Value representing the control requested for the physical or logical output.

Range is –32768 to +32767. Where the commanded value is unknown, this field shall contain 0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—A point index in object group 43 corresponds to the same physical or logical point as the identical index in object groups 40, 41, and 42.

NOTE 2—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 41 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.151 Analog output command event—Single-precision, floating-point without time

DNP3 Object Library		Group:	43
		Variation:	5
Group Name:	Analog Output Command Event	Type:	Event
Variation Name:	Single-precision, floating-point without time		

Description

See object group 43, variation 1 for a description of Analog Output Command Events and event detection.

Variation 5 objects contain a status octet that resulted when the outstation processed the requested control, and the single-precision floating-point commanded value. Information provided in this object relates to a control request, and not to the resultant state or status of the point controlled.

Coding

Pictorial

Formal structure

UINT7: Status Code

This value represents the status from processing the command that this event object is reporting. An enumeration of status codes appears in 11.1.

BSTR1: Reserved

Always 0

FLT32: Commanded Value

The Commanded Value representing the control requested for the physical or logical output.

Range is approximately -3.4×10^{38} to $+3.4 \times 10^{38}$. Where the commanded value is unknown, this field shall contain 0.0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

NOTE 1—See 11.1 for a description of the floating-point format.

NOTE 2—A point index in object group 43 corresponds to the same physical or logical point as the identical index in object groups 40, 41, and 42.

NOTE 3—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 41 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.152 Analog output command event—Double-precision, floating-point without time

DNP3 Object Library		Group:	43
		Variation:	6
Group Name:	Analog Output Command Event	Type:	Event
Variation Name:	Double-precision, floating-point without time		

Description

See object group 43, variation 1 for a description of Analog Output Command Events and event detection.

Variation 6 objects contain a flag octet and a double-precision floating-point value.

Coding

Pictorial

Formal structure

UINT7: Status Code

This value represents the status from processing the command that this event object is reporting. An enumeration of status codes appears in 11.1.

BSTR1: Reserved

Always 0

FLT64: Commanded Value

The Commanded Value representing the control requested for the physical or logical output.

Range is approximately -1.7×10^{308} to $+1.7 \times 10^{308}$. Where the commanded value is unknown, this field shall contain 0.0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

NOTE 1—See 11.1 for a description of the floating-point format.

NOTE 2—A point index in object group 43 corresponds to the same physical or logical point as the identical index in object groups 40, 41 and 42.

NOTE 3—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 41 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.153 Analog output command event—Single-precision, floating-point with time

DNP3 Object Library		Group:	43
		Variation:	7
Group Name:	Analog Output Command Event	Type:	Event
Variation Name:	Single-precision, floating-point with time		

Description

See object group 43, variation 1 for a description of Analog Output Command Events and event detection.

Variation 7 objects contain a flag octet, a single-precision floating-point value, and a time-of-occurrence.

Coding

Pictorial

Formal structure

UINT7: Status Code

This value represents the status from processing the command that this event object is reporting. An enumeration of status codes appears in 11.1.

BSTR1: Reserved

Always 0

FLT32: Commanded Value

The Commanded Value representing the control requested for the physical or logical output.

Range is approximately -3.4×10^{38} to $+3.4 \times 10^{38}$. Where the commanded value is unknown, this field shall contain 0.0.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—See 11.1 for a description of the floating-point format.

NOTE 2—A point index in object group 43 corresponds to the same physical or logical point as the identical index in object groups 40, 41, and 42.

NOTE 3—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 41 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.154 Analog output command event—Double-precision, floating-point with time

DNP3 Object Library		Group:	43
		Variation:	8
Group Name:	Analog Output Command Event	Type:	Event
Variation Name:	Double-precision, floating-point with time		

Description

See object group 43, variation 1 for a description of Analog Output Change Events and event detection.

Variation 8 objects contain a flag octet, a double-precision floating-point value, and a time-of-occurrence.

Coding

Pictorial

Formal structure

UINT7: Status Code

This value represents the status from processing the command that this event object is reporting. An enumeration of status codes appears in 11.1.

BSTR1: Reserved

Always 0

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

FLT64: Commanded Value

The Commanded Value representing the control requested for the physical or logical output.

Range is approximately -1.7×10^{308} to $+1.7 \times 10^{308}$. Where the commanded value is unknown, this field shall contain 0.0.

UINT48: Time-of-occurrence

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—See 11.1 for a description of the floating-point format.

NOTE 2—A point index in object group 43 corresponds to the same physical or logical point as the identical index in object groups 40, 41, and 42.

NOTE 3—This object group's events are assigned to a specific event class using an ***assign class*** message with objects from object group 41 and the respective indexes.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.155 Time and date—Absolute time

DNP3 Object Library		Group:	50
		Variation:	1
Group Name:	Time and Date	Type:	Info
Variation Name:	Absolute time		

Description

A Time and Date object with time only is an information type object that represents the absolute time.

This object is used to set or get the current time in an outstation.

Coding

Pictorial

Formal structure

UINT48: Absolute time value

Time expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—This object is referenced, but the object itself does not appear in a read request. The request is transmitted using qualifier code 0x07 and a range field value of 1. When an outstation receives this request, it implicitly indicates that the master wants the outstation to return the current time. The read response does contain a single Time and Date object; it is transmitted using qualifier code 0x07 and a range field value of 1.

NOTE 2—This object can be included in a write request. The request is transmitted using qualifier code 0x07 and a range field value of 1. When an outstation receives this request, it implicitly indicates that the master wants to set the current time in the outstation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.156 Time and date—Absolute time and interval

DNP3 Object Library		Group:	50
		Variation:	2
Group Name:	Time and Date	Type:	Info
Variation Name:	Absolute time and interval		

Description

A Time and Date object is an information object that represents the absolute time and a time interval.

This object is an information object that is used to specify an initial time and the time between subsequent, regular actions or activities. Function code 11 and 12 freeze commands use this object to specify when and how often counters or other objects should be frozen.

Coding

Pictorial

Formal structure

UINT48: Absolute time value

Time expressed as the number of milliseconds since the start of January 1, 1970. This is when the initial action (freeze in the case of function code 11 and 12) should commence.

UINT32: Interval time value

Time, expressed in milliseconds, between when periodic actions (freezes in the case of function code 11 and 12) should occur after the initial occurrence.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.157 Time and date—Absolute time at last recorded time

DNP3 Object Library		Group:	50
		Variation:	3
Group Name:	Time and Date	Type:	Info
Variation Name:	Absolute time at last recorded time		

Description

A Time and Date object with time only is an information type object that represents the absolute time when the most recently transmitted Record Current Time function code (24) was sent.

This object is used in LAN applications to set the current time in an outstation.

Coding

Pictorial

Formal structure

UINT48: Absolute time value

Time expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—This object is included in a write request. The request is transmitted using qualifier code 0x07 and a range field value of 1. When an outstation receives this request, it implicitly indicates that the master wants to set the current time in the outstation.

NOTE 2—The time value in this object is the absolute time when the last bit of the last octet in the most recently transmitted Record Current Time function code (24) was sent

NOTE 3—When this object is used, there is an implicit assumption that the propagation delay in the LAN is insignificant.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.158 Time and date common time-of-occurrence—Absolute time, synchronized

DNP3 Object Library		Group:	51
		Variation:	1
Group Name:	Time and Date Common Time-of-Occurrence	Type:	Info
Variation Name:	Absolute time, synchronized		

Description

A Time and Date Common Time-of-Occurrence object is an information type object that provides the absolute time as a reference from which other objects that follow can use to compute their absolute time. The relative time value included with a subsequent object is added to, or subtracted from, the time in this object in order to calculate the absolute time associated with the subsequent object.

This object may only appear in a response when the time in the outstation is believed to be accurately synchronized with a known time reference. The synchronization can be with respect to the DNP3 master, another DNP3 process in the outstation, or a local source—the choice is system dependent.

Use group 51, variation 2 objects when the time in the outstation has never been set or when the elapsed time since the last time update from the synchronizing source has exceeded a predetermined, device-specific limit.

Coding

Pictorial

Formal structure

UINT48: Absolute time value

Time expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—Time and Date Common Time-of-Occurrence is often abbreviated to Time and Date CTO.

NOTE 2—The primary reason for transmitting a Time and Date Common Time-of-Occurrence object followed by data objects with relative time offsets is to save bandwidth when the times associated with the data objects are closely spaced.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

NOTE 3—An example of an object that depends on a Time and Date Common Time-of-Occurrence object is a binary input change event with relative time, object group 2, variation 3.

NOTE 4—The following shows how multiple Time and Date CTO objects may be included in a response when there are not enough bits in a data object to hold the relative time with respect to a single Time and Date CTO object. Each data object's time is relative to the immediately preceding Time and Date CTO. In the figure, the time in DO_{i+1} is relative to $T\&D_1$:

$T\&D_0$	DO_0	DO_1	•••	DO_i	$T\&D_1$	DO_{i+1}	DO_{i+2}	•••

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.159 Time and date common time-of-occurrence—Absolute time, unsynchronized

DNP3 Object Library		Group:	51
		Variation:	2
Group Name:	Time and Date Common Time-of-Occurrence	Type:	Info
Variation Name:	Absolute time, unsynchronized		

Description

A Time and Date Common Time-of-Occurrence object is an information type object that provides the absolute time as a reference from which other objects that follow can use to compute their absolute time. The relative time value included with a subsequent object is added to, or subtracted from, the time in this object in order to calculate the absolute time associated with the subsequent object.

This object may only appear in a response when the time in the outstation has never been set or when the elapsed time since the last time update from the synchronizing source has exceeded a predetermined, device-specific limit.

Use group 51, variation 1 objects in a response when the time in the outstation is believed to be accurately synchronized with a known time reference. The synchronization can be with respect to the DNP3 master, another DNP3 process in the outstation, or a local source—the choice is system dependent.

Coding

Pictorial

Formal structure

UINT48: Absolute time value

Time expressed as the number of milliseconds since the start of January 1, 1970.

NOTE 1—Time and Date Common Time-of-Occurrence is often abbreviated to Time and Date CTO.

NOTE 2—The primary reason for transmitting a Time and Date Common Time-of-Occurrence object followed by data objects with relative time offsets is to save bandwidth when the times associated with the data objects are closely spaced.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

NOTE 3—An example of an object that depends on a Time and Date Common Time-of-Occurrence object is a binary input change event with relative time, object group 2, variation 3.

NOTE 4—The following shows how multiple Time and Date CTO objects may be included in a response when there are not enough bits in a data object to hold the relative time with respect to a single Time and Date CTO object. Each data object's time is relative to the immediately preceding Time and Date CTO. In the figure, the time in DO_{i+1} is relative to $T\&D_1$:

$T\&D_0$	DO_0	DO_1	•••	DO_i	$T\&D_1$	DO_{i+1}	DO_{i+2}	•••

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.160 Time delay—Coarse

DNP3 Object Library		Group:	52
		Variation:	1
Group Name:	Time Delay	Type:	Info
Variation Name:	Coarse		

Description

This object is used to convey the elapsed time between two absolute times with a resolution of 1 second.

Coding

Pictorial

Formal structure

UINT16: Delay seconds

This is the elapsed time with a 1-second resolution.

Range is 0 to +65 535 seconds.

NOTE—Use of the Fine Time Delay object, group 52, variation 2 is preferred for delay times of less than or equal to 65 seconds.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.161 Time delay—Fine

DNP3 Object Library		Group:	52
		Variation:	2
Group Name:	Time Delay	Type:	Info
Variation Name:	Fine		

Description

This object is used to convey the elapsed time between two absolute times with a resolution of 1 millisecond.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
									b0	Delay milliseconds
b15										

Formal structure

UINT16: Delay milliseconds

This is the elapsed time with a 1-millisecond resolution.

Range is 0 to +65 535 milliseconds.

NOTE—Use the Coarse Time Delay object, group 52, variation 1 for delay times of greater than 65 seconds.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.162 Class objects—Class 0 data

DNP3 Object Library		Group:	60
		Variation:	1
Group Name:	Class Objects	Type:	Info
Variation Name:	Class 0 data		

Description

A Class 0 Data object is an object placeholder that is transmitted in a ***read*** request to specify that the outstation return all of the data from specific static object groups as shown in the following table:

Object group	Description
1	Binary Input
3	Double-bit Binary Input
10	Binary Output
20	Counter
21	Frozen Counter
30	Analog Input
31	Frozen Analog Input
40	Analog Output Status

There are other restrictions regarding the inclusion of frozen and non-frozen counter and analog input data in the same response. See 11.1 for more details.

Other data may be included if specified in the Application Layer document or 11.1.

Coding

A Class 0 Data object does not carry any information in itself and, therefore, does not have an object coding.

NOTE 1—This object may only be used in ***read*** requests.

NOTE 2—The associated qualifier field in the request should always specify all objects (code 0x06) because there is no means to identify specific objects without explicitly including their object groups in the message.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.163 Class objects—Class 1 data

DNP3 Object Library		Group:	60
		Variation:	2
Group Name:	Class Objects	Type:	Info
Variation Name:	Class 1 data		

Description

A Class 1 Data object is an object placeholder that is transmitted in a ***read*** request to specify that the outstation return some or all objects having a class 1 event attribute. When an outstation receives a request with this object, it shall return class 1 event objects from its event queue or queues if it has more than one queue.

Coding

A Class 1 Data object does not carry any information in itself and, therefore, does not have an object coding.

NOTE 1—This object may only be used in ***read*** requests.

NOTE 2—The associated qualifier field in the request may specify all objects (code 0x06), or it may specify a count of objects (codes 0x07 or 0x08).

NOTE 3—See the Application Layer document section that discusses event reporting order.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.164 Class objects—Class 2 data

DNP3 Object Library		Group:	60
		Variation:	3
Group Name:	Class Objects	Type:	Info
Variation Name:	Class 2 data		

Description

A Class 2 Data object is an object placeholder that is transmitted in a ***read*** request to specify that the outstation return some or all objects having a class 2 event attribute. When an outstation receives a request with this object, it shall return class 2 event objects from its event queue or queues if it has more than one queue.

Coding

A Class 2 Data object does not carry any information in itself and, therefore, does not have an object coding.

NOTE 1—This object may only be used in ***read*** requests.

NOTE 2—The associated qualifier field in the request may specify all objects (code 0x06), or it may specify a count of objects (codes 0x07 or 0x08).

NOTE 3—See the Application Layer document section that discusses event reporting order.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.165 Class objects—Class 3 data

DNP3 Object Library		Group:	60
		Variation:	4
Group Name:	Class Objects	Type:	Info
Variation Name:	Class 3 data		

Description

A Class 3 Data object is an object placeholder that is transmitted in a ***read*** request to specify that the outstation return some or all objects having a class 3 event attribute. When an outstation receives a request with this object, it shall return class 3 event objects from its event queue or queues if it has more than one queue.

Coding

A Class 3 Data object does not carry any information in itself and, therefore, does not have an object coding.

NOTE 1—This object may only be used in ***read*** requests.

NOTE 2—The associated qualifier field in the request may specify all objects (code 0x06), or it may specify a count of objects (codes 0x07 or 0x08).

NOTE 3—See the Application Layer document section that discusses event reporting order.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.166 File-control—File identifier—superceded

DNP3 Object Library		Group:	70
		Variation:	1
Group Name:	File-Control	Type:	Info
Variation Name:	File identifier - Superceded		

Description

File Identifier objects have been superceded. The DNP3 Application Layer document discusses appropriate function codes and replacement objects.

Brief documentation of this object appears below for reference. Readers who need to know more details should consult the original DNP3 documentation.

Coding

Pictorial

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Formal structure

UINT16: Name size

Number of octets in the File name string element.

UINT8: File type code

0:	8-bit binary
1:	8-bit ASCII
2:	7-bit ASCII
3:	EBCDIC
4:	BCD
5:	5-bit Baudot
6:	6-bit Baudot

UINT8: Attribute code

0:	Regular or real permanent file.
1:	Temporary file
2:	Directory
3:	FIFO (first-in-first-out) File or pipe

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT16: Start record

This is the first record associated with certain file function codes.

UINT16: End record

This is the last record associated with certain file function codes.

UINT32: File size

Total number of octets in the file.

UINT48: Time of creation

Time expressed as the number of milliseconds since the start of January 1, 1970, when the file was created or last modified.

BSTR16: Permission

Bit 0:	World Execute
Bit 1:	World Write
Bit 2:	World Read
Bit 3:	Group Execute
Bit 4:	Group Write
Bit 5:	Group Read
Bit 6:	Owner Execute
Bit 7:	Owner Write
Bit 8:	Owner Read
Bits 9–15:	Reserved, always 0

UINT32: File ID

Unique identifier for the file.

UINT32: Owner ID

Unique identifier for the file owner.

UINT32: Group ID

Unique identifier for the file owner's group.

UINT8: File function code

0:	Append
1:	Delete
2:	Insert
3:	Write
4:	Erase
5:	Information
6:	Change working directory

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

7:	Present working directory
8:	Execute
9:	Read
10–254:	Not used.
255:	Response

UINT8: Status code

0:	Successful
1:	Does not exist
2:	Out of space
3:	Permission denied
4:	Busy

OSTRn: File name string

Name of file where the n in OSTRn is the number of octets specified by the name size element.

UINT16: First, second … and last record size

This is the number of octets in the respective data record that immediately follows.

OSTRm: First, second … and last record data octets

These are the contents of the record where the m in OSTRm is the number of octets specified by the respective record size element.

NOTE—New DNP3 implementations should avoid using this object.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.167 File-control—Authentication

DNP3 Object Library		Group:	70
		Variation:	2
Group Name:	File-Control	Type:	Info
Variation Name:	Authentication		

Description

An Authentication object is used for a master to request an authorization key from an outstation for use with subsequent file-control transactions.

Coding

Pictorial

Formal structure

UINT16: User name string offset

This is the offset (number of octets) from the beginning of the object to the first octet of the user name string in the object.

UINT16: User name string size

In a master request, this is the number of octets in the user name string appearing in the object.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Outstations always set this value to 0 in responses to indicate there is no user name string appearing in the returned object.

UINT16: Password string offset

This is the offset (number of octets) from the beginning of the object to the first octet of the password string in the object.

UINT16: Password string size

In a master request, this is the number of octets in the password string appearing in the object.

Outstations always set this value to 0 in responses to indicate there is no password string appearing in the returned object.

UINT32: Authentication key

This value shall be 0 in a master request message.

Outstation responses return a 32-bit number for use with future file-control transactions. A non-zero value indicates specific permissions have been granted, whereas a 0 value indicates world or guest permissions.

OSTRn: User name string

Masters include a non-null-terminated user name string in request messages that indicates the name of the user requesting an authentication key. The n in OSTRn indicates the number of octets in the user name string.

For security reasons, outstations do not return a user name—this element contains zero octets in responses.

OSTRm: Password string

Masters include a non-null-terminated password string in request messages that indicates the user's password. The m in OSTRm indicates the number of octets in the password string.

For security reasons, outstations do not return a password—this element contains zero octets in responses.

NOTE—The authentication key returned is valid for only a single transaction unless otherwise stated in other DNP3 documentation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.168 File-control—File command

DNP3 Object Library		Group:	70
		Variation:	3
Group Name:	File-Control	Type:	Info
Variation Name:	File command		

Description

A File Command object is used with Application Layer function codes 25 and 27 to initiate open and delete operations.

Coding

Pictorial

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Formal structure

UINT16: File name string offset

This is the offset (number of octets) from the beginning of the object to the first octet of the file name string at the end of the object.

UINT16: File name string size

In a master request, this is the number of octets in the file name string appearing in the object.

Outstations always set this value to 0 in responses to indicate there is no user name string appearing in the returned object.

UINT48: Time of creation

Time expressed as the number of milliseconds since the start of January 1, 1970, when the file was created or last modified.

This field should contain a time value when opening a file for writing (see operational mode below). For other operations and when the time of creation is unknown, this field should contain a 0.

BSTR16: Permissions

Bit 0:	World Execute
Bit 1:	World Write
Bit 2:	World Read
Bit 3:	Group Execute
Bit 4:	Group Write
Bit 5:	Group Read
Bit 6:	Owner Execute
Bit 7:	Owner Write
Bit 8:	Owner Read
Bits 9–15:	Reserved, always 0

Permission is determined by the logical ORing of the bits. The relevant action is permitted if the bit is set and inhibited if the bit is clear. (0x1FF indicates no restrictions.)

This field should contain a non-zero value when opening a file for writing (see operational mode below). For other operations, this field should contain a 0.

UINT32: Authentication key

Masters enter the value returned from an authentication transaction immediately preceding the request that contains this object.

A non-zero value indicates specific permissions have been granted, whereas a 0 value indicates world or guest permissions. Enter 0 if the outstation does not support authentication.

UINT32: File size

Total number of octets in the file.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

This field should contain a non-zero value when opening a file for writing or appending (see operational mode below). This provides a means for the outstation to know how much buffer or storage space it may need to allocate for the entire file after the writing is completed. For other operations, this field should contain a 0.

A file size of 0xFFFFFFFF indicates that the actual file size is unknown. Outstation devices are not required to accept unknown file sizes and may reject the request.

UINT16: Operational mode

This field applies to file open requests and specifies how the outstation should prepare itself for subsequent file control operations. The following table indicates allowable codes for this field.

Code	Mode name	Description
0	NULL	This code is used for non-open command requests.
1	READ	Specifies that an existing file is to be opened for reading.
2	WRITE	Specifies that the file is to be opened for writing. If the file already exists, this specifies that the existing file should be over-written and its length shall be truncated to 0 before writing the data octets.
3	APPEND	Specifies a file shall be opened for writing and that octets are to be appended to the end of that file. If the file does not already exist, it shall be created. Minimum implementations are not obligated to support this mode.
all others	Reserved	These codes are reserved for future use.

UINT16: Maximum block size

This item is used for file open commands. It is used to negotiate the number of file octets transferred in subsequent read or write request messages to an amount that is satisfactory for both the master and the outstation.

If the file is opened for reading, this value sets a maximum limit on the number of file data octets that the master shall accept in a single response. The outstation may send no more than this number of octets from the file in each g70v5 object. Outstations may send fewer octets in the response.

If the file is opened for writing, this value sets the maximum limit on the number of octets from the file that the master shall send in each g70v5 object. The master may be required to transport fewer octets than this value in subsequent write requests if the outstation's response indicates a lower number in its *maximum block size* field—see File Command Status object, g70v4.

UINT16: Request ID

This is a master-dependent parameter that the outstation is required to return in the *request ID* field of its corresponding response. It permits the master to associate a response with a particular request.

Typically, the master obtains this value from a single running counter that it maintains for all file-control requests to all outstations. The count is incremented with each new request. This approach is not a DNP3 requirement, and masters are free to choose any scheme, including no scheme, of their choosing.

Outstations do not interpret or deduce any meaning from the value in this field.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

OSTRn: File name string

Masters include a non-null-terminated, file name string in request messages that indicates the name of the file to which the action specified by the Application Layer function code applies. The n in OSTRn indicates the number of octets in the file name string.

NOTE—The authentication key is valid for only a single transaction unless otherwise stated in other DNP3 documentation.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.169 File-control—File command status

DNP3 Object Library		Group:	70
		Variation:	4
Group Name:	File-Control	Type:	Info/Event
Variation Name:	File command status		

Description

A File Command Status object is used in file-control responses and requests, depending on the Application Layer function code in the request.

Responses

This object appears in all responses to open, close, delete, and abort operations, Application Layer function codes 25, 26, 27, and 30. It is also used for responses to get file information commands, Application Layer function code 28, when the outstation needs to report an error condition. When the outstation sends this object as an event, it shall request Application Layer confirmation.

Requests

This object appears in requests associated with closing a file or aborting an operation, Application Layer function codes 26 and 30.

Coding

Pictorial

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Formal structure

UINT32: File handle

This parameter provides a unique numeric identifier for an opened file. Its value is assigned by the outstation based on the outstation's own file management algorithm. The file handle is valid only as long as the file remains opened. The handle is transmitted in future transaction requests from the master that opened the file to identify the same file.

An outstation should never specify the same file handle value for more than one master at a time.

When this object is included in the response to a request for opening a file, the file handle shall be memorized by the master for future file-control transactions only if the *status code* indicates success; it has no meaning if the operation is unsuccessful.

When this object is used to responses to non-open requests, the file handle specifies which file the outstation acted, or attempted to act, upon.

When this object is used for file-control requests, such as close and abort, the file handle specifies which file the outstation should act upon.

UINT32: File size

This field only applies to responses returned from an outstation when a master requests to open a file for reading. See File Command object, group 70, variation 3. In this case, the value is the total number of octets in the file.

For all other transactions involving a File Command Status object, the file size is set to 0.

UINT16: Maximum block size

This item is returned in responses to file open commands. For all other transactions involving a File Command Status object, the file size is set to 0. It is used to negotiate the number of file octets transferred in subsequent read or write messages to an amount that is satisfactory for both the master and the outstation.

If the file is to be opened for reading, this value specifies the maximum number of file octets that the outstation shall include in a single response. Outstations may send fewer file octets in the subsequent read response messages. This value shall be less than or equal to the *maximum block size* value that the master specified in the File Command object, group 70, variation 3, requesting to open the file.

If the file is to be opened for writing, this value sets the maximum limit on the number of octets from the file that the outstation can accept in a single write request. The master may send fewer file octets in the subsequent write request messages. When this value is less than the *maximum block size* value that the master specified in the File Command object, group 70, variation 3, requesting to open the file, then the master shall limit the number of file octets it transmits to this lower value.

UINT16: Request ID

When this object is included in a request message, the outstation is required to return in the same value in the *request ID* field of its corresponding response. It permits the master to associate a response with a particular request.

When this object is returned in a response message, this value shall match the *request ID* value found in the file-control object associated with the request.

Outstations do not interpret or deduce any meaning from the value in this field.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Additional information is provided in the description of the *request ID* field in the File Command object, group 70, variation 3.

UINT8: Status code

The status code indicates whether the request was successful, or if not, it gives an appropriate error code to explain what went wrong.

File-control status codes are enumerated in the DNP3 Object Library document. Of those, only codes 0 through 9 and 255 apply to a File Command Status object.

OSTRn: Optional text

This field provides a means for an outstation to include optional text explanations that supplement a terse *status code* when an error is detected. DNP3 outstations are encouraged to fill in this field, but they are not required to do so. DNP3 masters shall accept responses with characters in this field; however, they are not obligated to do anything with this information.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.170 File-control—File transport

DNP3 Object Library		Group:	70
		Variation:	5
Group Name:	File-Control	Type:	Info/Event
Variation Name:	File transport		

Description

A File Transport object is used to transmit octets from a file between a master and an outstation in either direction.

Requests

A File Transport object is used in file-control requests to read or write a block of data from the file, Application Layer function codes 1 and 2. When this object appears in such a request, it is considered an Information type object. When the outstation sends this object as an event, it shall request Application Layer confirmation.

Responses

This object is used in responses to requests to read a block of data from a file, Application Layer function code 1, only if the read operation is successful. When the outstation shall respond immediately, this object is considered an Information type object; however, if the outstation returns the file octets in a delayed response, the object is considered to be an Event type object.

Coding

Pictorial

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Formal structure

UINT32: File handle

In a request, this parameter is the file handle returned from the outstation during a file open transaction, and it specifies which file the outstation shall act upon.

In a response, this parameter indicates to which file the response pertains.

UINT31: Block number

This field specifies the block number associated with the group of data octets from the file.

The block numbers are initialized to zero when a file is opened and incremented by one count after each subsequent transfer of file octets. The first block transferred has a block number of 0.

BSTR1: Last

This bit is set in the last block of the file transfer and cleared in all other blocks.

OSTRn: File data octets

These are the file's octets. The most number of octets in any message is dependent on the maximum block size negotiated during the open transaction.

NOTE 1—Blocks do not need to be the same size for each block transferred and can vary from message-to-message. Therefore, no assumptions should be made pertaining to an offset within the file being a fixed multiple of the block number.

FileOffset = N * BlockNumber; <- This is wrong!

NOTE 2—Both the outstation and the master should update the file offset after every each read or write transfer.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.171 File-control—File transport status

DNP3 Object Library		Group:	70
		Variation:	6
Group Name:	File-Control	Type:	Info/Event
Variation Name:	File transport status		

Description

A File Transport object is used to report the status of a file transport operation.

An outstation returns this object in a response to a read request, Application Layer function code 1, only if the read operation was **not** successful.

An outstation always returns this object a response to a write request, Application Layer function code 2.

When the outstation responds immediately, this object is considered an Information type object; however, if the outstation returns the object in a delayed response, the object is considered to be an Event type object. When the outstation sends this object as an event, it shall request Application Layer confirmation.

Coding

Pictorial

Formal structure

UINT32: File handle

This parameter indicates to which file the response pertains.

UINT31: Block number

This field specifies the block number associated with the group of data octets from the file.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

BSTR1: Last

This bit is set in the last block of the file transfer and cleared in all other blocks.

UINT8: Status code

The status code indicates whether the request was successful, or if not, it gives an appropriate error code to explain what went wrong.

File-control status codes are enumerated in 11.1. Of those, only codes 0, 6, 8, 16 through 20, and 255 apply to a File Transport Status object.

OSTRn: Optional text

This field provides a means for an outstation to include optional text explanations that supplement a terse *status code* when an error is detected. DNP3 outstations are encouraged to fill in this field, but they are not required to do so. DNP3 masters shall accept responses with characters in this field, however; they are not obligated to do anything with this information.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.172 File-control—File descriptor

DNP3 Object Library		Group:	70
		Variation:	7
Group Name:	File-Control	Type:	Info/Event
Variation Name:	File descriptor		

Description

Requests

A File Descriptor object is used in file-control requests with Application Layer function code 29 to obtain information about the file named in the object. When this object appears in a request, it is considered an Information type object.

Responses

This object is used in responses to requests for file information, Application Layer function code 29, only if the operation to get the file information is successful. When the outstation responds immediately, this object is considered an Information type object (see the right-hand column of the heading at top of this page); however, if the outstation returns the information in a delayed response, the object is considered to be an Event type object. When the outstation sends this object as an event, it shall request Application Layer confirmation.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
									b0	File name string offset
b15										
									b0	File name string size
b15										
									b0	File type
b15										
									b0	File size
b31										
									b0	Time of creation
b47										
	OW	OE	GR	GW	GE	WR	WW	WE	b0	Permissions
b15	0	0	0	0	0	0	0	OR		

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Formal structure

UINT16: File name string offset

This is the offset (number of octets) from the beginning of the object to the first octet of the file name string at the end of the object.

UINT16: File name string size

This is the number of octets in the file name string appearing in the object.

UINT16: File type

This value is set to 0 in requests.

In a response, this field specifies the type of file per the following chart.

Type	Description
0	File is a directory.
1	File is a simple file that is suitable for sequential file transfer.
all others	These codes are reserved for future use.

UINT32: File size

This value is set to 0 in requests.

In responses:

For file type 0 files, directories, this is the number of entries in the subdirectory, excluding any links to itself or its parent directory.

For file type 1 files, this is the total number of octets in the file.

A file size of 0xFFFFFFFF indicates that the actual file size is unknown.

UINT48: Time of creation

This value is set to 0 in requests.

In a response this is the time expressed as the number of milliseconds since the start of January 1, 1970, when the file was created or last modified.

BSTR16: Permissions

This value is set to 0 in requests.

In responses:

Bit 0: World Execute

Bit 1: World Write

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Bit 2:	World Read
Bit 3:	Group Execute
Bit 4:	Group Write
Bit 5:	Group Read
Bit 6:	Owner Execute
Bit 7:	Owner Write
Bit 8:	Owner Read
Bits 9–15:	Reserved, always 0

Permission is determined by the logical ORing of the bits. The relevant action is permitted if the bit is set and inhibited if the bit is clear. (0x1FF indicates no restrictions.)

UINT16: Request ID

When this object is included in a request message, the outstation is required to return in the same value in the *request ID* field of its corresponding response. It permits the master to associate a response with a particular request.

When this object is returned in a response message, this value shall match the *request ID* value found in the File Descriptor object associated with the request.

Outstations do not interpret or deduce any meaning from the value in this field.

Additional information is provided in the description of the *request ID* field in the File Command object, group 70, variation 3.

OSTRn: File name string

Masters include a non-null-terminated, file name string in request messages that indicates the name of the file about which information is desired. The n in OSTRn indicates the number of octets in the file name string.

Outstations include a non-null-terminated, file name string in response messages to indicate the name of the file about which the information applies.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.173 File-control—File specification string

DNP3 Object Library		Group:	70
		Variation:	8
Group Name:	File-Control	Type:	Info
Variation Name:	File specification string		

Description

This object is used to identify a specific file using its name and possibly the complete path to the file.

This is a variable length object whose size depends on the number of characters required to uniquely identify the file.

This object may only be used with qualifier 0x5B because different files may use a different number of characters depending on their location, the vendor's file system, and which vendor supplies the device. If there is an index number associated with a file, it is not referenced in the object headers that specify object group 70, variation 8.

Coding

Pictorial

The value of *size* in the file specification string is the number of characters in the string. This value would also appear in the prefix appearing before the object.

Formal structure

OSTRn: File specification string

File specification strings do not use a null-terminator. All characters have non-zero values.

The left-most character is transmitted first. For example, if the file name were "Config.bin," the "C" character would appear in the octet 0 position.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

NOTE 1—DNP3 does not specify whether files named in this object use short file names, such as

MyConfig.bin

or fully qualified file names that include a path to the file, such as

\RTU\ConfigurationFiles\Analog.cfg.

It is the responsibility of devices using this object to support a length and naming style that is required to uniquely identify the intended file.

NOTE 2—It is recommended to keep the names as short as possible.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.174 Internal Indications—Packed format

DNP3 Object Library		Group:	80
		Variation:	1
Group Name:	Internal Indications	Type:	Static
Variation Name:	Packed format		

Description

Object group 80, variation 1 is used to report and in some cases set a device's DNP3 Internal Indication states. Internal indications are described in detail in Clause 4 through Clause 6 of this standard. internal indication states are reported in every response from an outstation. The internal indications reported or set by this object group and variation are the same.

Point indexes are used to specify the specific internal indication bit(s) according to the following table.

Index	Bit	Name
0	IIN1.0	ALL_STATIONS
1	IIN1.1	CLASS_1_EVENTS
2	IIN1.2	CLASS_2_EVENTS
3	IIN1.3	CLASS_3_EVENTS
4	IIN1.4	NEED_TIME
5	IIN1.5	LOCAL_CONTROL
6	IIN1.6	DEVICE_TROUBLE
7	IIN1.7	DEVICE_RESTART
8	IIN2.0	NO_FUNC_CODE_SUPPORT
9	IIN2.1	OBJECT_UNKNOWN
10	IIN2.2	PARAMETER_ERROR
11	IIN2.3	EVENT_BUFFER_OVERFLOW
12	IIN2.4	ALREADY_EXECUTING
13	IIN2.5	CONFIG_CORRUPT
14	IIN2.6	RESERVED_1
15	IIN2.7	RESERVED_2

A device may include a private set of internal indication states beyond index 15.

Coding

This object is coded as a bit string. When multiple indexes are contained in a single message, they are packed, as illustrated in the following pictorial.

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
	7	6	5	4	3	2	1	0	b0	Internal Indication States
b15	15	14	13	12	11	10	9	8		

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

This figure depicts the bit-packing sequence for a message containing bit indexes 0 through 15. As can be seen, the first bit is in the bit 0 position of the first octet, the second bit is in the bit 1 position of the first octet, etc. Any unused bits in the final octet are returned as 0.

Formal structure

BSTRn: Internal Indication States

n in BSTRn represents the number of indexes reported in the bit string.

Bit 0 in the bit string corresponds to the lowest point index. Subsequent bits correspond to sequentially increasing point indexes.

Bit 0 in the bit string is aligned with bit 0 of the first octet.

Unused bits in the last octet are padded with 0.

Each bit has a value of 0 or 1, representing the internal indication state.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.175 Device storage—Buffer fill status

DNP3 Object Library		Group:	81
		Variation:	1
Group Name:	Device Storage	Type:	Info
Variation Name:	Buffer fill status		

Description

A Device Storage—Buffer Fill Status object conveys what percentage of internal buffer space is filled.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position	
OV									Overflow state and fill percent
									Object group
									Variation

Formal structure

UINT7: Fill percentage

Percentage of allocated space that is currently filled. Range is 0 to 100.

BSTR1: Overflow state.

0: Normal.

1: Overflowed.

UINT8: Group

The group number along with the variation field specify which buffer's status is indicated.

UINT8: Variation

The variation number along with the group field specify which buffer's status is indicated.

NOTE—This object is **NOT** recommended for new DNP3 implementations. Originally, this object was conceived to indicate how much of the memory allocated for queues, buffers, and other storage facilities were currently occupied. However, the application of this object is heavily dependent on the software architecture of the outstation. Each vendor is free to implement DNP3 in a manner that is best for his or her organization, and because of this, it is not possible to provide uniformity for masters. Masters would need to provide customize software for each vendor and possibly each product produced by the same vendor.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.176 Device profile—Functions and indexes

DNP3 Object Library		Group:	82
		Variation:	1
Group Name:	Device Profile	Type:	Info
Variation Name:	Functions and indexes		

Description

This device profile object provides a means for an outstation to tell the master which functions it supports and what indexes for each object group.

A master does not request this object; instead an outstation has the option of sending it when the request from the master is not recognizable, un-parsable, or objects referenced in the request are not supported. The Internal Indications octet indicates the problem in parsing, and this object provides information so that the master (or its operators) can determine how to adjust future requests.

Upon receipt of this object, the master (or its operators) can change the polling scheme, poll request message(s), limit or expand the assumed functionality of the outstation, or re-configure the database.

A device profile object has two sections. The first specifies which Application Layer function codes are supported by the outstation. The second section specifies what range of indexes is valid for specific object groups.

Coding

Pictorial

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Group	
Variation	Last object header
Qualifier	
Range	

Formal structure

BSTR64: Function codes 0 to 63 supported.

Each bit corresponds to an Application Layer function code by position. For example, bit 1 (second bit) aligns with function code 1, READ. Set bits indicate supported functions. Function codes higher than 63 are reported by appending a similar BSTR64 that should be appended to this object; the bits correspond to function codes 64 through 127.

UINT16: Count of object headers that follow.

This value specifies the number of object headers that follow.

Object Headers: N sets of object headers, where N is the count specified previously.

Each object header consists of the standard DNP3 group, variation, qualifier, and range octets.

Valid qualifier codes are as follows:

— 07 and 08. The range field contains the count of the indexes supported by the outstation—points are assumed to have contiguous index numbers beginning at 0.

— 00 and 01. The range field contains the start and stop indexes.

Each object header may use any one of the valid qualifier codes—the headers do not need to use the same qualifiers.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.177 Data set—Private registration object

DNP3 Object Library		Group:	83
		Variation:	1
Group Name:	Data Set	Type:	Any
Variation Name:	Private registration object		

Description

A Private Registration Object (PRO) object contains a uniquely defined grouping of data. The specific group of data is private but known to the master and outstation communicating the PRO object.

Example uses for PRO objects are as follows:

— Snapshot of values associated with a power transformer, including voltages, currents, power factors, oil temperature, ambient temperature, and tap setting.

— Harmonic currents in phases A, B, and C of an electrical feeder.

— Periodically recorded upstream pressure, downstream pressure, temperature, and orifice size for a gas pipe.

— Tuning parameters associated with a PID loop.

A specific PRO object may be reported as static or event data or as information depending on the system in which it is employed. The application may also determine whether the PRO may be included with class 1, 2, or 3 events or with class 0 static data or whether the object shall be polled for separately from class data and whether the object may be included in an unsolicited transmission.

Each PRO object is matched one-to-one with a Private Registration Object Descriptor (g83v2) object that describes the structure and format of the data contained within the PRO.

Coding

Pictorial

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Formal structure

VSTR4: Vendor Code

This is a four-octet vendor name field. Vendors should select a unique acronym for their company.

UINT16: Object identifier

A number that uniquely identifies the data set. This field is also known as a Private Registration Number or PRN. It is a private number that allows devices to distinguish one data set from another.

UINT16: Length

The total number octets in all of the data objects that follow this field.

Data objects:

The data objects are specified in the corresponding Private Registration Object Descriptor having the same vendor and Object identifier (PRN).

NOTE 1—The objects in a data set may have indexes for the respective point type they are associated with; however, this is not necessary. There is no means using the data object specifiers in the Private Registration Object Descriptor to correlate an object in a PRO with an index.

NOTE 2—Applications may optionally choose to include a single time object in the data set to specify the time at which all the other objects were recorded.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.178 Data set—Private registration object descriptor

DNP3 Object Library		Group:	83
		Variation:	2
Group Name:	Data Set	Type:	Info
Variation Name:	Private registration object descriptor		

Description

A Private Registration Object Descriptor (PROD) object describes the structure and format of a uniquely defined grouping of data that is private to the outstation. See object g83v1 for more information about the data sets associated with Private Registration Objects.

Each PROD object is matched one-to-one with a Private Registration Object (g83variation) object.

This object may be used for determination of how to parse a Private Registration Object.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
									b0	Vendor code
b31										
									b0	Object identifier
b15										
									b0	Count of data object specifiers
b15										
									b0	(1st) Object quantity
b15										
b7									b0	(1st) Object group
b7									b0	(1st) Object variation
									b0	(2nd) Object quantity
b15										
b7									b0	(2nd) Object group
b7									b0	(2nd) Object variation
										●●●
									b0	(nth) Object quantity
b15										
b7									b0	(nth) Object group
b7									b0	(nth) Object variation

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Formal structure

VSTR4: Vendor Code

This is a four-octet vendor name field. Vendors should select a unique acronym for their company.

UINT16: Object identifier

A number that uniquely identifies the data set. This field is also known as a Private Registration Number or PRN. It is private number that allows devices to distinguish one data set from another.

UINT16: Count of data object specifiers

The number (N) of data object specifier sets that follow this field.

N sets of data object specifiers:

Each specifier corresponds to a data set, or group of data having the same point type.

Format of data object specifier

UINT16: Object quantity

The number of objects having the object group and variation specified by the following two octets.

UINT8: Object group

The object group for objects in this set.

UINT8: Object variation

The object variation for objects in this set.

NOTE—The objects in a data set may have indexes for the respective point type they are associated with; however, this is not necessary. There is no means using the data object specifiers to correlate an object in a PRO with an index.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.179 Data set prototype—With UUID

DNP3 Object Library		Group:	85
		Variation:	1
Group Name:	Data Set Prototype	Type:	Info
Variation Name:	With UUID		

Description

This object is used to transport a list of descriptor elements that define a group or subset of elements that exist in a data set. References to prototypes may appear in a data set descriptor as a shorthand method for indicating that the data set shall contain a subset of elements described by the lists of descriptors in the prototypes.

A data set descriptor consists of a sequence of descriptor elements that include a length, a descriptor code, a data type code, a maximum data length, and possibly an ancillary value.

Coding

Pictorial

Formal structure

SET of n: Descriptor elements

{

UINT8: Length.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Specifies the total number of octets in the *descriptor code*, *data type code*, *maximum data length*, and *ancillary value* fields. If there is no ancillary value field, the length is 3.

UINT8: Descriptor code.

Specifies what type of descriptor element this is.

UINT8: Data type code.

Specifies the type of data in the value field of the corresponding data set element. If there is not a corresponding element in the data set, such as descriptor elements having descriptor codes of ID, UUID, NSPC, and NAME, this field should specify data type code NONE.

UINT8: Maximum data length.

Specifies the maximum number of octets the value field may contain for the corresponding element in the data set. For example, if the data code were UINT, and the maximum data length were set to 4, then the value shall be represented by a 1-, 2-, 3-, or 4-octet (8-, 16-, 24- or 32-bit) unsigned integer. If the data code were VSTR, and the maximum data length were set to 32, the string length in the data set's element shall not exceed 32 characters.

If there is not a corresponding element in the data set, such as descriptor elements having descriptor codes of ID, UUID, NSPC, and NAME, this field should specify 0.

Variant: Ancillary value. (Variant implies dependency on the descriptor element's data descriptor code and length fields.)

Ancillary value that depends on the descriptor code.

}

NOTE 1—The first descriptor element should be an ID element. It contains an identifier for the prototype. This identifier is not related to the identifiers in data sets or data set descriptors. It is used to access individual prototypes using classic DNP3 index numbers.

NOTE 2—The second descriptor element in a data set prototype should be a UUID element. It is the UUID associated with the prototype. If any member of an existing prototype, except the ID element, should need to change in any respect, the new prototype shall have a new UUID, including changes to names, namespaces, types, maximum lengths, and so forth.

NOTE 3—If prototype namespace and name are used, they should appear in the third and fourth descriptor elements; the third element shall be a NSPC element, and the fourth should be a NAME. These refer to the namespace and name of the prototype, not to the individual or specific data set with which it is associated.

NOTE 4—Both NSPC and NAME shall appear or both should not. It is not permissible to have a NSPC without a NAME or a NAME without a NSPC.

NOTE 5—Each data set descriptor should fit within a single Application Layer fragment.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.180 Data set descriptor—Data set contents

DNP3 Object Library		Group:	86
		Variation:	1
Group Name:	Data Set Descriptor	Type:	Info
Variation Name:	Data set contents		

Description

This object describes the sequence of elements in a data set and the data types of the values in the data set.

A data set descriptor consists of a sequence of descriptor elements that include a length, a descriptor code, a data type code, a maximum data length, and possibly an ancillary value.

Coding

Pictorial

Formal structure

SET of n: Descriptor elements

{

UINT8: Length.

Specifies the total number of octets in the *descriptor code*, *data type code*, *maximum data length*, and *ancillary value* fields. If there is no ancillary value field, the length is 3.

UINT8: Descriptor code.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Specifies what type of descriptor element this is.

UINT8: Data type code.

Specifies the type of data in the value field of the corresponding data set element. If there is not a corresponding element in the data set, such as descriptor elements having descriptor codes of ID, NAME, and PTYP, this field should specify data type code NONE.

UINT8: Maximum data length.

Specifies the maximum number of octets the value field may contain for the corresponding element in the data set. For example, if the data code were UINT, and the maximum data length were set to 4, then the value shall be represented by a 1-, 2-, 3-, or 4-octet (8-, 16-, 24-, or 32-bit) unsigned integer. If the data code were VSTR, and the maximum data length were set to 32, the string length in the data set's element shall not exceed 32 characters.

If there is not a corresponding element in the data set, such as descriptor elements having descriptor codes of ID, NAME, and PTYP, this field should specify 0.

Variant: Ancillary value. (Variant implies dependency on the descriptor element's data descriptor code and length fields.)

Ancillary value that depends on the descriptor code.

}

NOTE 1—Each data set descriptor should fit within a single Application Layer fragment.

NOTE 2—For each outstation–master pair of devices, event and static data sets that relate to the same collection[42] of data are based on the same data set descriptor and have the same identifier as appears in the data set descriptor.

[42] The term "collection" is used in this document as a noun to represent a group or set of data. It was chosen to avoid confusion with other terms that have a specific meaning in DNP3.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.181 Data set descriptor—Characteristics

DNP3 Object Library		Group:	86
		Variation:	2
Group Name:	Data Set Descriptor	Type:	Info
Variation Name:	Characteristics		

Description

This object conveys the transmission attributes of a data set. The attributes indicate

— Whether the data set is readable and/or writable.

— Whether static and/or event variations exist.

— Which device, master or outstation, defined the data set.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
0	0	0	DF	EV	ST	WR	RD	Characteristics

Formal structure

BSTR4: Characteristics

Bit 0:	RD—set if data set is readable.
Bit 1:	WR—set if data set is writable.
Bit 2:	ST—set if outstation maintains a static data set.
Bit 3:	EV—set if outstation generates a data set event.
Bit 4:	DF—set if defined by master, and cleared if defined by outstation.

NOTE 1—The index number associated with this object corresponds to the identifier in a data set descriptor, object group 86, variation 1.

NOTE 2—This object may not be returned in response to a read request that specifies object group 86, variation 0; the request should specify variation 2.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.182 Data set descriptor—Point index attributes

DNP3 Object Library		Group:	86
		Variation:	3
Group Name:	Data Set Descriptor	Type:	Info
Variation Name:	Point index attributes		

Description

This object conveys the point indexes corresponding to the data elements in a data set.

Coding

Pictorial

Formal structure

UINT8: Length.

Specifies the number of octets in the *data set identifier* field.

UINTn: Data set identifier.

This is a number that uniquely identifies the point index set. This number provides correspondence to a data set descriptor and the resultant data sets.

SET of n: Point index elements

{

UINT8: Length.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Specifies the number of octets in the *point type* and *point index* fields.

UINT8: Point type.

Specifies the point type with which the index is associated. Point type values are the group number used to convey static data.

UINTn: Point index.

An index number relative to the point indexes used to convey data values for the specified point type in traditional DNP3 read or write requests.

}

NOTE 1—Data set point index attribute objects are transmitted in their entirety. It is not permitted to send portions of the object; the entire object should fit within a single Application Layer fragment.

NOTE 2—The order of the point index elements should match the order appearing in the data set descriptor having the same identifier as that found in the *data set identifier* field.

NOTE 3—If a value in the data set is not correlated to a specific point in the classic database, group number 0 is used.

NOTE 4—This object may not be returned in response to a read request that specifies object group 86, variation 0; the request should specify variation 3.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.183 Data set—Present value

DNP3 Object Library		Group:	87
		Variation:	1
Group Name:	Data Set	Type:	Static
Variation Name:	Present value		

Description

This object conveys the present or most recently obtained values in a data set.

Coding

Pictorial

Formal structure

UINT8: Length.

Specifies the number of octets in the *data set identifier* field.

UINTn: Data set identifier.

This is a number that uniquely identifies the data set. This number provides correspondence to a data set descriptor.

UINT8: Length.

Specifies the number of octets in the *evaluation time* field. This value is always 6.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT48: Evaluation time.

The evaluation time is the time when a snapshot of all the element values were obtained or computed and buffered for transmission. This time is not adjusted if a repeated response is sent.

It is expressed as the number of milliseconds since the start of January 1, 1970.

SET of n: Data elements.

{

UINT8: Length.

Specifies the number of octets in the *value* field.

Variant: Value. (Variant implies dependency on the data type and the element's *length* field.)

Data element's value. The number octets shall not exceed the limit specified in the *maximum data length* field of the corresponding descriptor element.

}

NOTE 1—The first four values—length, data set identifier, length, and evaluation time—are mandatory even though there is not a corresponding descriptor element in the corresponding data set descriptor. All of the element values that follow, in the set of n, are transmitted in the same order as they appear in the corresponding data set descriptor object.

NOTE 2—Data sets are transmitted in their entirety. It is not permitted to send portions of a data set. Data sets shall fit within a single Application Layer fragment.

NOTE 3—The evaluation time is an absolute time and has no interrelationship to other DNP3 times that appear in other objects in the same response message.

NOTE 4—The format of the data that appears in each of the value fields is specified by the data type code in the corresponding data set descriptor element. The data type only specifies the format, integer, floating-point, visible string, etc. The number of octets in the value is specified by the *length* field and may not exceed the limit specified in the *maximum data length* field of the corresponding descriptor element.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.184 Data set event—Snapshot

DNP3 Object Library		Group:	88
		Variation:	1
Group Name:	Data Set Event	Type:	Event
Variation Name:	Snapshot		

Description

This object conveys a data set event.

Coding

Pictorial

Formal structure

UINT8: Length.

Specifies the number of octets in the *data set identifier* field.

UINTn: Data set identifier.

This is a number that uniquely identifies the data set. This number provides correspondence to a data set descriptor.

UINT8: Length.

Specifies the number of octets in the *time-of-event* field. This value is always 6.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT48: Time-of-event.

Time when the event occurred expressed as the number of milliseconds since the start of January 1, 1970.

SET of n: Data elements.

{

UINT8: Length.

Specifies the number of octets in the *value* field.

Variant: Value. (Variant implies dependency on the data type and the element's *length* field.)

Data element's value. The number octets shall not exceed the limit specified in the *maximum data length* field of the corresponding descriptor element.

}

NOTE 1—The first four values—length, data set identifier, length, and time of event—are mandatory even though there is not a corresponding descriptor element in the corresponding data set descriptor. All of the element values that follow, in the set of n, are transmitted in the same order as they appear in the corresponding data set descriptor object.

NOTE 2—Data sets are transmitted in their entirety. It is not permitted to send portions of a data set. Data sets should fit within a single Application Layer fragment.

NOTE 3—The time-of-event is an absolute time and has no interrelationship to other DNP3 times that appear in other objects in the same response message. For example, consider a response that contains a common-time-of-occurrence object, group 51, some binary input events objects, and a data set event object. The data set event object should still include the time element, and that time is not affected or influenced by the previous common-time-of-occurrence object.

NOTE 4—The format of the data that appears in each of the value fields is specified by the data type code in the corresponding data set descriptor element. The data type only specifies the format, integer, floating-point, visible string, etc. The number of octets in the value is specified by the *length* field and may not exceed the limit specified in the *maximum data length* field of the corresponding descriptor element.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.185 Application—Identifier

DNP3 Object Library		Group:	90
		Variation:	1
Group Name:	Application	Type:	Info
Variation Name:	Identifier		

Description

An Application Identifier object is used to represent an application or operating system process within a device. It is used in conjunction with application-related function codes (INTIALIZE_APPL, START_APPL, and STOP_APPL) to control applications.

Coding

Pictorial

Not applicable. See formal structure.

Formal structure

The format of this object is privately defined by the device where the application resides.

NOTE—Object qualifier code 0x5B should be used with this object if a specific application is referenced in the message.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.186 Status of requested operation—Active configuration

DNP3 Object Library		Group:	91
		Variation:	1
Group Name:	Status of Requested Operation	Type:	Info
Variation Name:	Activate configuration		

Description

This object provides a sequence of status elements that indicate the fitness or suitability of the corresponding objects in an ***activate configuration*** (Application Layer function code 31) request or the configuration data referenced by objects in the request.

This status results from the checking performed prior to the outstation's actually attempting to activate the configuration.

Coding

Pictorial

Formal structure

UINT32: Time delay.

The number of milliseconds during which the outstation expects to be busy and shall not respond to requests.

This value may be 0 if an error is detected and the outstation shall not attempt to activate the configuration. It may also be 0 if the outstation shall not restart and can accept new requests immediately.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINT8: Number of status elements.

Specifies the total number of status elements in this object. There shall be one status element for each object in the request, and the ordering of the status elements shall correspond to the order in which objects appear in the request.

SET of n: Status elements

{

UINT8: Length.

Specifies the total number of octets in the *status code* and *ancillary text* fields. If there is no ancillary text field, the length is 1.

UINT8: Status code.

Set to 0 if the outstation does not detect any errors in the corresponding request object or in the configuration data referenced by the corresponding request object.

Set to 1 if the outstation detects an error in the request object. One example is that the outstation is unable to locate a file referenced by a file specification string, g70v8. Another example is that the outstation does not have a name referenced by an octet string, g110, object.

Set to 2 if the outstation detects an error in the configuration data referenced by the corresponding request data.

Set to 3 for any other error.

Set to 4 if not checked.

VSTRn: Ancillary text.

Any text that elaborates on the status code. This text may be logged at the master station to assist in diagnosing problems.

Examples for status code 1 are “File not found” or “Unknown name.”

Examples for status code 2 are “Scale factor exceeds limits.”

}

NOTE—If the request contains multiple objects, and an error is detected in any one of them, it is permissible to stop the error checking immediately without checking the remaining objects. The status code for the unchecked objects should be reported as 4.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.187 Floating-point—None—general description common to all variations

DNP3 Object Library		Group:	100
		Variation:	All
Group Name:	Floating-Point	Type:	Static
Variation Name:	None—General Description Common to All Variations		

Description

Floating-point objects are obsolete. New applications should use the Analog Input types, groups 30 through 33.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.188 Binary-coded decimal integer—Small

DNP3 Object Library		Group:	101
		Variation:	1
Group Name:	Binary-Coded Decimal Integer	Type:	Static
Variation Name:	Small		

Description

This object is used to convey 4-digit, binary-coded decimal values for which other objects are not appropriate.

Coding

Pictorial

Formal structure

BCD4: BCD value

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.189 Binary-coded decimal integer—Medium

DNP3 Object Library		Group:	101
		Variation:	2
Group Name:	Binary-Coded Decimal Integer	Type:	Static
Variation Name:	Medium		

Description

This object is used to convey 8-digit, binary-coded decimal values for which other objects are not appropriate.

Coding

Pictorial

Formal structure

BCD8: BCD value

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.190 Binary-coded decimal integer—Large

DNP3 Object Library		Group:	101
		Variation:	3
Group Name:	Binary-Coded Decimal Integer	Type:	Static
Variation Name:	Large		

Description

This object is used to convey 16-digit, binary-coded decimal values for which other objects are not appropriate.

Coding

Pictorial

Formal structure

BCD16: BCD value

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.191 Unsigned integer—8-bit

DNP3 Object Library		Group:	102
		Variation:	1
Group Name:	Unsigned Integer	Type:	Static
Variation Name:	8-bit		

Description

This object is used to convey single-byte values for which other objects are not appropriate.

This object can be used for reading and writing memory locations within a device using virtual address qualifier codes.

Coding

Pictorial

octet transmission order ↓

7	6	5	4	3	2	1	0	← bit position
b8							b0	Byte value

Formal structure

UINT8: Byte value.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.192 Octet string—None—general description common to all variations

DNP3 Object Library		Group:	110
		Variation:	All
Group Name:	Octet String	Type:	Static
Variation Name:	None—General Description Common to All Variations		

Description

This description applies to all Octet String, group 110, variations. See 11.1 for a description of an Octet String Point Type.

Octet strings may be used to represent any block of 8-bit quantities.

The variation used with an Octet String is the length of the string. A response may contain a variation smaller than requested if the available string length is shorter than what was requested.

Coding

Pictorial

The following figure illustrates an Octet String of n octets (variation n).

Formal structure

OSTRn: Value

This is the most recently measured, obtained, or computed value.

Range is 0 to 255 within each octet.

The octet string length, denoted as the n in OSTRn, is the same as the variation number, and may not exceed 255.

NOTE 1—Octet String objects are not included in class 0 polls.

NOTE 2—Reading and writing of 8-bit memory locations can be implemented using this object together with absolute addressing qualifiers.

NOTE 3—Only ***read***, ***write***, and ***response*** function codes are permitted.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.193 Octet string event—None—general description common to all variations

DNP3 Object Library		Group:	111
		Variation:	All
Group Name:	Octet String Event	Type:	Event
Variation Name:	None—General Description Common to All Variations		

Description

This description applies to all Octet String Event, group 111, variations. See 11.1 for a description of an Octet String Point Type.

The variation used with an octet string is the length of the string. A response may contain a variation smaller than requested if the available string length is shorter than what was requested.

Coding

Pictorial

The following figure illustrates an Octet String Event of n octets (variation n).

Formal structure

OSTRn: Value

This is the most recently measured, obtained, or computed value of the Octet String at the time when the event was generated.

Range is 0 to 255 within each octet.

The octet string length, denoted as the n in OSTRn, is same as the variation number, and may not exceed 255.

NOTE—Only ***read***, ***response***, and ***unsolicited response*** function codes are permitted.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.194 Virtual terminal output block—None—general description common to all variations

DNP3 Object Library		Group:	112
		Variation:	All
Group Name:	Virtual Terminal Output Block	Type:	Static
Variation Name:	None—General Description Common to All Variations		

Description

This description applies to all Virtual Terminal Output Block, group 112, variations. See 11.1 for a description of a Virtual Terminal Point Type.

The variation used with a Virtual Terminal Output Block is the length of the string.

Coding

Pictorial

The following figure illustrates a Virtual Terminal Output Block object of n octets (variation n).

Formal structure

OSTRn: Terminal data

Each octet can have a value that ranges from 0 to 255. Exclusive use of ASCII characters is not required.

The octet string length, denoted as the n in OSTRn, is the same as the variation number, and may not exceed 255.

The content of the data in the object is specific to the particular terminal protocol. DNP3 masters and outstations do not need to know any of the details of the terminal protocol; DNP3 does not assign any significance to any of the data octets in this object.

NOTE 1—Objects of this type are not to be returned in a class 0 poll.

NOTE 2—The Device Profile Document should list the point index number(s) assigned to virtual terminals.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.195 Virtual terminal event data—None—general description common to all variations

DNP3 Object Library		Group:	113
		Variation:	All
Group Name:	Virtual Terminal Event Data	Type:	Event
Variation Name:	None—General Description Common to All Variations		

Description

This description applies to Virtual Terminal Event Data, group 113, all variations. See 11.1 for a description of a Virtual Terminal Point Type.

The variation used with a Virtual Terminal Event Data is the length of the string.

Coding

Pictorial

The following figure illustrates a Virtual Terminal Event Data object of n octets (variation n).

Formal structure

OSTRn: Terminal data

Each octet can have a value that ranges from 0 to 255. Exclusive use of ASCII characters is not required.

The octet string length, denoted as the n in OSTRn, is the same as the variation number, and may not exceed 255.

The content of the data in the object is specific to the particular terminal protocol. DNP3 masters and outstations do not need to know any of the details of the terminal protocol; DNP3 does not assign any significance to any of the data octets in this object.

NOTE 1—The Device Profile Document should list the point index number(s) assigned to virtual terminals.

NOTE 2—This object may be assigned to class 1, 2, or 3.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.196 Authentication—Challenge

DNP3 Object Library		Group:	120
		Variation:	1
Group Name:	Authentication	Type:	Info
Variation Name:	Challenge		

Description

This object is used for DNP3 secure authentication as described in Clause 7. This object may be included in either a request or a response. When this object appears alone in the request or response, its appearance requires that the other station (either master or outstation) transmit an Authentication Reply object with the HMAC Value calculated based on this object and on the DNP3 fragment the other station most recently transmitted.

If the outstation transmits this object at the end of a response with the CON bit set, the outstation is requesting that the master transmit the application Confirm using aggressive mode authentication (g120v3 and g120v9).

Coding

Pictorial

Formal structure

UINT32: Challenge Sequence Number (CSQ).

Devices shall use this value to match responses with challenges. Each device shall maintain its own separate Challenge Sequence number. Each device shall set this value to zero on startup and increment this value each time it transmits a Challenge message. If the value reaches 4 294 967 295, the next CSQ the Challenger transmits shall be zero.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Challenge sequence numbers shall be independent of User Number. In other words, each device need only store a single value of CSQ locally regardless of how many users or devices it is communicating with.

UINT16: User Number (USR)

The responder shall use this value to identify which set of Session Keys is to be used in this challenge-response sequence.

<0> := Unknown. The challenge-response sequence is being initiated by an outstation. Therefore the appropriate USR is not yet known. The master shall supply the appropriate USR in the Authentication Reply.

<1> := Default. The challenge-response sequence is being initiated by a master on behalf of more than one user, and the set of Session Keys used shall therefore be the default set of keys for this master-outstation pair. Refer to Clause 7 for more information.

<2…65535> := Chosen by the master station to be associated with a particular user and corresponding set of Session Keys.

UINT8: HMAC algorithm (HAL).

Using this value, the Challenger shall specify the algorithm that the Responder shall use to calculate the HMAC Value, and shall also specify the resulting length of the HMAC Value. Refer to Clause 7 for details on how to calculate these algorithms.

<0> := not used

<1> := HMAC SHA-1 truncated to 4 octets (serial)

<2> := HMAC SHA-1 truncated to 10 octets (networked)

<3> := HMAC SHA-256 truncated to 8 octets (serial)

<4> := HMAC SHA-256 truncated to 16 octets (networked)

<5..127> := reserved for future use

<128..255> := reserved for vendor-specific choices. Not guaranteed to be interoperable.

IMPORTANT: Refer to the note in the supplement regarding the dependency between the use of truncated HMAC algorithms and the need for frequent Session Key changes. In any case, the longest practical HMAC should be used; the shorter options are only provided to address performance issues on bandwidth-limited systems.

UINT8: Reason for Challenge.

This value explains the Challenger's reason for making the challenge. The Responder shall use this value to determine what extra data to include when calculating the HMAC Value.

<0> := not used

<1> := CRITICAL. Challenging a critical function. The Responder shall include the entire *previous* ASDU transmitted by the Responder when calculating the HMAC Value, as well as any further protocol-specific information.

<2..255> := reserved for future use

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINTn: Pseudo-Random Challenge Data.

Devices shall include pseudo-random data in the Challenge message to verify that the contents of the Challenge message are not predictable. The pseudo-random data shall be generated using the algorithm specified in the FIPS 186-2 Digital Signature Standard. The minimum length of the Challenge Data shall be 4 octets.

NOTE—In the DNP3 implementation of IEC/TS 62351-5 authentication, the length of the Challenge Data is not specified within the object but in the Object Prefix. The Authentication Challenge object should always be used with a Qualifier of 0x5B, indicating that the object is of variable length up to 65 535 octets, specified in the Object Prefix. The length of the Challenge Data is therefore the size specified in the Object Prefix, minus 8 octets.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.197 Authentication—Reply

DNP3 Object Library		Group:	120
		Variation:	2
Group Name:	Authentication	Type:	Info
Variation Name:	Reply		

Description

This object is used for DNP3 secure authentication as described in Clause 7. This object may be included in either a request or a response. This shall be the only object to appear in the request or response. It is a reply to an Authentication Challenge object (g120v1).

Coding

Pictorial

Formal structure

UINT32: Challenge sequence number (CSQ).

The value transmitted in this object shall be the same value transmitted by the Challenger in the Authentication Challenge object.

UINT16: User Number (USR).

The sender shall use this value to identify which set of Session Keys the Challenger should use to authenticate this Reply. If the sender is the outstation, this value shall be the same as the USR value transmitted by the master in the previous Authentication Challenge message. If the sender is the master, it shall set the USR value according to which user is being authenticated. Refer to the discussion in Clause 7 for more information.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINTn: HMAC value.

The sender shall calculate the HMAC Value according to the HMAC algorithm specified by the Challenger, as described in Clause 7. The sender shall include in the HMAC Value calculation the data listed in Table A.3, in the order listed.

Table A.3—Data included in the HMAC value calculation

Data	Description	Included
Challenge message	The entire DNP3 Application Layer fragment containing the Authentication Challenge object.	Always.
Addressing information	Although IEC/TS 62351-5 specifies a protocol may include addressing information from lower layers, DNP3 authentication does not include any such addresses.	Never.
Challenged ASDU	The entire DNP3 Application Layer fragment being challenged, including the Application Layer header.	If the Reason For Challenging is <1>, challenging a critical function.
Padding Data	Any padding data required.	As required by the HMAC algorithm.

Outstations sending this object shall use the current Monitoring Direction Session Key to calculate the HMAC Value.

Masters sending this object shall use the current Control Direction Session Key to calculate the HMAC value.

NOTE—In the DNP3 implementation of IEC/TS 62351-5 authentication, the length of the HMAC Value is not specified within the object but in the Object Prefix. The Authentication Reply object should always be used with a Qualifier of 0x5B, indicating that the object is of variable length up to 65 535 octets, specified in the Object Prefix. The length of the HMAC Value is therefore the size specified in the Object Prefix, minus 6 octets.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.198 Authentication—Aggressive mode request

DNP3 Object Library		Group:	120
		Variation:	3
Group Name:	Authentication	Type:	Info
Variation Name:	Aggressive Mode Request		

Description

This object is used for DNP3 secure authentication as described in Clause 7. This object may be included as the first object in either a DNP3 request or a DNP3 response. It attempts to authenticate the fragment it appears in. Using aggressive mode is a less secure means of authentication than issuing an Authentication Challenge object and receiving an Authentication Reply. However, it requires much less bandwidth.

Aggressive mode shall be preceded by a Challenge and a Reply. A device shall not transmit an Aggressive Mode Request until the device has received at least one Challenge message from the Challenger. Refer to the procedures in the *Secure Authentication* supplement for more details.

The initial Challenge is necessary because the sender of this object uses the data from the most recently received Challenge message to calculate the Challenge Sequence Number and the HMAC in the Aggressive Mode Request.

As shown in Figure A.1, the device shall include an Authentication HMAC object (g120v9) as the last object in the same fragment as the Authentication Aggressive Mode Request object.

Application Header	Object Header g120v3 Authentication Aggressive Mode Request	Object g120v3 Authentication Aggressive Mode Request	Object headers and objects to be authenticated, e.g., g12v1 Control Relay Output Block	Object Header g120v9 Authentication HMAC	Object g120v9 Authentication HMAC

Figure A.1—Fragment using aggressive mode authentication

Coding

Pictorial

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Formal structure

UINT32: Challenge sequence number (CSQ).

The Challenge Sequence Number (CSQ) shall have the value described in the definition of the Authentication Challenge object (g120v1). The effect of the rules described in that section is that the CSQ of a given Aggressive Mode Request shall be the CSQ from the most recently received Challenge message, plus the number of Aggressive Mode Requests that the sender of this object has transmitted since receiving that Challenge message, plus one.

UINT16: User Number

The sender of this object shall use this value to identify which set of Session Keys is to be used to authenticate this Aggressive Mode Request.

<0> := Unknown. Not used for this object.

<1> := Default. One of two cases is occurring:

— This object is being sent by a master on behalf of more than one user.

— This object is being sent by an outstation and there is no corresponding user.

In either case, the set of Session Keys used shall be the default set used for this master–outstation pair. Refer to Clause 7 for more information.

<2…65535> := Chosen by the master station to be associated with a particular user and corresponding set of Session Keys.

NOTE—The DNP3 device should transmit this object using the qualifier 0x07 (single-octet count of objects) with a count of 1.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.199 Authentication—Session key status request

DNP3 Object Library		Group:	120
		Variation:	4
Group Name:	Authentication	Type:	Info
Variation Name:	Session Key Status Request		

Description

This object is used for DNP3 secure authentication as described in Clause 7. This object may be included in a DNP3 request sent by the master only. This shall be the only object to appear in the request. The function code to be used in this request is an Authentication Request (0x20). It is intended to elicit a DNP3 response with function code Authentication Response (0x83) containing a Session Key Status object.

Coding

Pictorial

octet transmission order ↓

	7	6	5	4	3	2	1	0	← bit position	
									b0	User Number
b15										

Formal structure

UINT16: User Number

The master shall use this value to identify which set of Session Keys it is querying.

<0> := Unknown. Not used for this object.

<1> := Default. The default set of Session Keys used by the master on behalf of multiple users, or used when the outstation initiates the sequence of messages that results in an authentication. Refer to Clause 7 for a description of when this value should be used.

<2…65535> := Chosen by the master station to be associated with a particular user and corresponding set of Session Keys.

NOTE—The master should use the qualifier 0x07 (one octet count of objects) in the object header for this object.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.200 Authentication—Session key status

DNP3 Object Library		Group:	120
		Variation:	5
Group Name:	Authentication	Type:	Info
Variation Name:	Session Key Status		

Description

This object is used for DNP3 secure authentication as described in Clause 7. This object is included in a DNP3 response only. This shall be the only object to appear in the response. It is transmitted in response to a Session Key Status Request object and represents the current state of the Session Keys as known by the outstation. If the outstation considers the Session Keys to be valid, this object is authenticated with an HMAC.

Coding

Pictorial

Formal structure

UINT32: Key Change Sequence Number (KSQ)

Each outstation shall maintain a Key Change sequence number, which it shall use to match Key Status messages with subsequent Key Change messages. This value shall be initialized to zero on startup of the outstation. The outstation shall increment the KSQ

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

each time it receives a Session Key Change or Session Key Status Request. (The first KSQ transmitted shall therefore always be 1.) If the value reaches 4 294 967 295, the next KSQ the outstation transmits shall be zero.

The master shall not process the KSQ except to include it in subsequent Key Change messages.

UINT16: User Number

The outstation shall use this value to identify the set of Session Keys for which it is reporting the current status. This value shall match the value supplied in the previous Session Key Status Request, as described in the definition of that object.

UINT8: Key wrap algorithm

Using this value, the outstation shall indicate to the master the algorithm it shall use to decrypt the data in subsequent Session Key Change objects. Each device shall support at least the minimum subset of algorithms listed in Clause 7.

<0> := not used

<1> := AES-128 Key Wrap Algorithm, as described in Clause 7.

<2..127> := reserved for future use

<128..255> := reserved for vendor-specific choices. Not guaranteed to be interoperable.

UINT8: Key status

This value describes the status of the two Session Keys as known by the outstation.

<0> := not used

<1> := OK. There have been no communications failures or restarts since the last time the outstation received an authentic Key Change message. The Session Keys are valid.

<2> := NOT_INIT. The outstation has not received an authentic Key Change message since it last started up, nor has not received such a message within the Session Key Change Interval or Session Key Change Count configured at the outstation. The Session Keys are not valid.

<3> := COMM_FAIL. The outstation has detected a communications failure in either the control or the monitoring direction. The Session Keys are not valid.

<4> := AUTH_FAIL. The outstation has received a non-authentic Challenge or Aggressive Mode Request. The Session Keys are not valid.

NOTE—This shall also be the response if the USR number supplied is invalid.

<5..255> := reserved for future use

UINT8: HMAC algorithm (HAL).

Using this value, the outstation shall specify the algorithm that the master shall use to calculate the HMAC Value, and shall also specify the resulting length of the HMAC Value. Refer to Clause 7 for details of how to calculate these algorithms.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

<0> := No HMAC Value in this message. The outstation shall use this value in all cases except when the Key Status = OK.

<1> := HMAC SHA-1 truncated to 4 octets (serial)

<2> := HMAC SHA-1 truncated to 10 octets (networked)

<3> := HMAC SHA-256 truncated to 8 octets (serial)

<4> := HMAC SHA-256 truncated to 16 octets (networked)

<5..127> := reserved for future use

<128..255 := reserved for vendor-specific choices. Not guaranteed to be interoperable.

IMPORTANT: Refer to the note in the supplement regarding the dependency between the use of truncated HMAC algorithms and the need for frequent Session Key changes.

UINT16: Challenge Data Length

This value defines the length of the challenge data that follows.

UINTn: Pseudo-random challenge data

The outstation shall include this pseudo-random data in the Key Status message to verify that the contents of the Key Status message are not predictable. The pseudo-random data shall be generated using the algorithm specified in the *FIPS 186-2 Digital Signature Standard*.

UINTn: HMAC value.

If the Key Status = OK, the outstation shall calculate the HMAC Value according to the HMAC algorithm HAL, as described in Clause 7. The outstation shall include in the HMAC Value calculation the data listed in Table A.4, in the order listed. It shall use the Monitoring Direction Session Key from the Session Key Change object most recently received from the master. If the Key Status is not = OK, there is no HMAC Value included in this object and HAL = 0.

Table A.4—Data included in the HMAC value calculation

Data	Description	Included
Challenge message	The entire DNP3 Application Layer fragment containing the Session Key Change object most recently received by the outstation.	Always.
Padding Data	Any padding data required.	As required by the HMAC algorithm.

NOTE—The Authentication Challenge object should always be used with a Qualifier of 0x5B, indicating that the object is of variable length up to 65 535 octets, specified in the Object Prefix. The length of the Challenge Data may therefore be either calculated from the qualifier and the length of the HMAC or read from the corresponding field of the object.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.201 Authentication—Session key change

DNP3 Object Library		Group:	120
		Variation:	6
Group Name:	Authentication	Type:	Info
Variation Name:	Session Key Change		

Description

This object is used for DNP3 secure authentication as described in Clause 7. The master performs an Authentication Request (0x20) of this object to supply the outstation with the Session Key that shall be used to calculate HMAC Values in all subsequent authentication operations. This shall be the only object to appear in the request.

The Session Keys are encrypted using a separate Update Key.

Coding

Pictorial

Formal structure

UINT24: Key Change Sequence Number (KSQ)

This value shall match the KSQ transmitted in the Session Key Status object most recently received by the master, as described in the definition of that object.

UINT16: User Number (USR)

The master shall use this value to specify which set of Session Keys is to be changed. It shall match the USR in the Session Key Status object most recently received by the master, as described in the definition of that object.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

UINTn: Wrapped key data

This value shall be the result of passing the Session Keys and the most recent Key Status message through the Key Wrap Algorithm defined in the Key Status message. The master shall pass the data through the Key Wrap Algorithm in the order described in Table A.5.

Table A.5—Data included in the key wrap (in order)

Data	Description	Included
Key Length	The size of one of the Session Keys. Both keys are the same length. This value is two octets long.	Always
Control Direction Session Key	The key used to authenticate data from the master.	Always
Monitoring Direction Session Key	The key used to authenticate data from the outstation.	Always
Key Status message	All data in the Session Key Status object most recently received from the outstation, KSQ first, not including any HMAC.	Always
Padding data	As required by the key wrap algorithm.	As required.

The Session Keys shall be treated as arrays of octets and transmitted with the lowest index octet first. For example, Appendix A of the AES specification provides the example of a 128-bit cipher key shown in Table A.6. The byte with index 0, having value 2b, shall be transmitted first.

Table A.6—Example of key order

Value	2b	7e	15	16	28	ae	d2	a6	ab	f7	15	88	09	cf	4f	3c
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Note that the output from the key wrap algorithm may be longer than the input. For instance, the AES Key Wrap Algorithm produces output that is exactly 8 octets longer than its input. Table A.7 shows a typical example of the Wrapped Key Data using this algorithm.

Table A.7—Example of wrapped key data

Data	Description	Size (in octets)
Key Length	So the outstation shall know what follows	2
Control Direction Session Key	Using the minimum size 128-bit keys	16
Monitoring Direction Session Key		16
Session Key Status object	Using the minimum size of challenge data, i.e., 4 octets	15
Padding data	Required to make the input data a multiple of 8 octets.	7
Additional output	For the AES key wrap algorithm	8
TOTAL		64

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

NOTE 1—In the DNP3 implementation of IEC/TS 62351-5 authentication, the length of the Wrapped Key Data is not specified within the object but in the Object Prefix. The Authentication Key Change object should always be used with a Qualifier of 0x5B, indicating that the object is of variable length up to 65 535 octets, specified in the Object Prefix. The length of the Wrapped Key Data is therefore the size specified in the Object Prefix, minus 6 octets.

NOTE 2—The outstation should respond to this request with a Session Key Status Object (g120v5).

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.202 Authentication—Error

DNP3 Object Library		Group:	120
		Variation:	7
Group Name:	Authentication	Type:	Event/Info
Variation Name:	Error		

Description

This object is used in DNP3 secure authentication as described in Clause 7. This object is used by a device (either master or outstation) to indicate an error with a previous authentication operation.

This object may be assigned to a Class and transmitted as an event object along with other data. The master may read the most recent event of this class for each user by reading the static Authentication Last Error objects (g120v8). If sent by a master, this object is considered of type Info rather than Event.

It is recommended that an instance of each Authentication Error event object be created on each DNP3 association to an outstation. Doing so permits the outstation to report security problems on one association to masters on other associations. Note that the User Number shall be unique within a DNP3 association, as described in the section of the supplement discussing data concentrators. Only the most recent Authentication Error event object shall be buffered; older event objects shall be discarded to avoid denial of service attacks.

Coding

Pictorial

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Formal structure

UINT32: Sequence Number

This value shall be the Challenge Sequence Number or Key Change Sequence Number, of the operation that the Error message is replying to.

UINT16: User Number (USR)

This value shall be the User Number of the operation that the Error message is replying to, identifying the set of Session Keys and Update Key in use. Note that the User Number may also be zero when the correct User Number is unknown. Refer to Clause 7.

UINT16: Association ID

This value shall uniquely identify the association between the master and the outstation on which the error occurred. The definition of a DNP3 association may be found in Clause 13. Because of the variety of configurations of DNP3 implementations, the Association ID may correspond to different combinations of DNP3 addresses, IP addresses, and port numbers or identifiers on the master and outstation. However, whatever mapping is used, the combination of User Number and Association ID shall be unique within the device.

UINT8: Error code.

This value shall specify the reason the error message is being transmitted.

<0> := not used

<1> := Authentication failed. The authentication information supplied by the other device was incorrect, or the data it was authenticating was corrupted in transit.

<2> := Unexpected response. The other device transmitted a message that did not follow the procedures as described in Clause 7.

<3> := No response. The other device either did not respond to the Challenge message or did not follow an Aggressive Mode Request with data for authentication.

<4> := Aggressive Mode not permitted. The device sending this Error Code does not permit the use of Aggressive Mode on this link.

<5> := HMAC algorithm not permitted. The device sending this Error Code does not permit the use of the specified HMAC algorithm on this link. Mandatory HMAC algorithms are specified in Clause 7.

<6> := Key Wrap algorithm not permitted. The device sending this Error Code does not permit the use of the specified Key Wrap algorithm on this link. Mandatory Key Wrap algorithms are specified in Clause 7.

<7> := Authorization failed. The authentication information supplied by the other device was correct, but the authenticated user is not permitted to perform the requested operation.

<8…127> := reserved for future standardization

<128..255> := private range for definition by each vendor. A device using this range shall use a different Error Code for each possible error reason, and shall supply an Error Text to explain each Error Code.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3TIME: Time of error.

Time when the event occurred expressed in standard DNP3 time.

UNCDn: Error text.

This value shall be a string of text suitable for display on a user interface or in a security log encoded in unicode UTF-8 as described in IETF RFC 3629 (note that all characters encoded in 7-bit ASCII comply with UTF-8). The Error Text shall explain the Error Code. It is recommended that the Error Text also include a unique description of the user represented by the User Number.

For standardized Error Codes, the Error Text is optional and the length of the text may be zero. For private range Error Codes, the Error Text shall be mandatory.

NOTE 1—In the DNP3 implementation of IEC/TS 62351-5 authentication, the length of the Error Text is not specified within the object but in the Object Prefix. The Authentication Error object should always be used with a Qualifier of 0x5B, indicating that the object is of variable length up to 65 535 octets, specified in the Object Prefix. The length of the Error Text is therefore the size specified in the Object Prefix, minus 15 octets.

NOTE 2—The Error Text is optional.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

A.1.2.203 Authentication—Hashed message authentication code (HMAC)

DNP3 Object Library		Group:	120
		Variation:	9
Group Name:	Authentication	Type:	Info
Variation Name:	Hashed Message Authentication Code (HMAC)		

Description

This object is used for DNP3 secure authentication as described in Clause 7. This object may be included as the last object in either a DNP3 request or a DNP3 response. It attempts to authenticate the fragment it appears in. Using aggressive mode is a less secure means of authentication than issuing an Authentication Challenge object and receiving an Authentication Reply. However, it requires much less bandwidth.

Aggressive mode shall be preceded by a Challenge. A device shall not transmit an Aggressive Mode Request until it has received at least one Challenge message. Refer to the procedures in the *Secure Authentication* supplement for more details.

The initial Challenge is necessary because the sender of this object uses data from the most recently received Challenge message to calculate the HMAC Value.

As shown in Figure A.2, the device shall include an Authentication Aggressive Mode Request object (g120v3) as the first object in the same fragment as the Authentication HMAC object.

Application Header	Object Header g120v3 Authentication Aggressive Mode Request	Object g120v3 Authentication Aggressive Mode Request	Object headers and objects to be authenticated, e.g., g12v1 Control Relay Output Block	Object Header g120v9 Authentication HMAC	Object g120v9 Authentication HMAC

Figure A.2—Fragment using aggressive mode authentication

Coding

Pictorial

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Formal structure

UINTn: HMAC Value

In aggressive mode, the HMAC Value shall be calculated in the same manner as in normal mode, but shall be calculated based on the *same* ASDU as the Aggressive Mode Request, rather than the *previous* ASDU. Table A.8 describes this difference.

Table A.8—Data included in the HMAC value calculation in aggressive mode

Data	Description	Included
Challenge message	The entire DNP3 Application Layer fragment containing the most recently received Authentication Challenge Object, including the CSQ at the time of that challenge.	Always.
Addressing information	Although IEC/TS 62351-5 specifies a protocol may include addressing information from lower layers, DNP3 authentication does not include any such addresses.	Never.
Authenticated Data	The entire DNP3 Application Layer fragment that this object is included in, including the Application Layer header, all objects preceding this one, and the object header for this object.	Always.
Padding Data	Any padding data required.	As required by the HMAC algorithm.

The length of the HMAC Value shall be determined by the HMAC algorithm (HAL) of the most recent Challenge received by the sender of the object, as described in the definition of the Authentication Challenge object.

Outstations sending this object shall use the current Monitoring Direction Session Key to calculate the HMAC Value.

Masters sending this object shall use the current Control Direction Session Key to calculate the HMAC Value.

NOTE—In the DNP3 implementation of IEC/TS 62351-5 authentication, the length of the HMAC Value is not specified within the object but in the Object Prefix. The Authentication Aggressive Mode Request object should be used with a Qualifier of 0x5B, indicating that the object is of variable length up to 65 535 octets, specified in the Object Prefix. The length of the HMAC Value is therefore the size specified in the Object Prefix.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Annex B

(informative)

DNP3 quick reference

Function Codes

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 resp	17	Delay measurement
7	Freeze	18	Record current time
8	Freeze – No resp	19	Open file
9	Freeze clear	1A	Close file
A	Freeze clear – No resp	1B	Delete file
B	Freeze at time	1C	Get file information
C	Freeze at time – No resp	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 error
Responses (Hex)			
81	Response	83	Authentication response
82	Unsolicited response		

Internal Indications

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

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Application Layer

Object Prefix

0	Objs packed without a prefix.
1	Objs prefixed with 1-octet index.
2	Objs prefixed with 2-octet index.
3	Objs prefixed with 4-octet index.
4	Objs prefixed with 1-octet object size.
5	Objs prefixed with 2-octet object size.
6	Objs prefixed with 4-octet object size.
7	Reserved.

Range Field Contains

0	1-octet start – stop indexes.
1	2-octet start – stop indexes.
2	4-octet start – stop indexes.
3	1-octet start – stop virtual addresses.
4	2-octet start – stop virtual addresses.
5	4-octet start – stop virtual addresses.
6	No range field used. Implies all values.
7	1-octet count of objects.
8	2-octet count of objects.
9	4-octet count of objects.
A	Reserved.
B	1-octet count of objects (variable format).
C	Reserved.
D	Reserved.
E	Reserved.
F	Reserved.

Transport Function

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Data Link Layer

Primary to secondary (PRM = 1)

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 to primary (PRM = 0)

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

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Valid Data Link Layer Control Codes

Outstation to Master	Master to Outstation	Function Code Name	Type	Comment
00	**80**	**ACK**	Sec-to-Pri	
01	**81**	**NACK**		Link reset required
0B	8B	LINK_STATUS		
0F	8F	NOT_SUPPORTED		
10	90	ACK		Receive buffers full
11	91	NACK		Receive buffers full
1B	9B	LINK_STATUS		Receive buffers full
1F	9F	NOT_SUPPORTED		Receive buffers full
40	**C0**	**RESET_LINK_STATES**	Pri-to-Sec	FCB = 0 (secondary ignores FCB)
44	**C4**	**UNCONFIRMED_USER_DATA**		FCB = 0 (secondary ignores FCB)
49	C9	REQUEST_LINK_STATUS		FCB = 0 (secondary ignores FCB)
52	D2	TEST_LINK_STATES		FCB = 0
53	**D3**	**CONFIRMED_USER_DATA**		FCB = 0
60	E0	RESET_LINK_STATES		FCB = 1 (secondary ignores FCB)
64	E4	UNCONFIRMED_USER_DATA		FCB = 1 (secondary ignores FCB)
69	E9	REQUEST_LINK_STATUS		FCB = 1 (secondary ignores FCB)
72	F2	TEST_LINK_STATES		FCB = 1
73	**F3**	**CONFIRMED_USER_DATA**		FCB = 1

Most commonly used are shown in **bold** face.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 Exchange Samples

Reset Link Example		
--►	05 64 05 C0 01 00 00 04 E9 21	Reset link states
◄--	05 64 05 00 00 04 01 00 19 A6	Ack

Integrity Poll Example		
--►	05 64 14 F3 01 00 00 04 0A 3B C0 C3 01 3C 02 06 3C 03 06 3C 04 06 3C 01 06 9A 12	Request class 1, 2, 3 and 0 data
◄--	05 64 05 00 00 04 01 00 19 A6	Link layer confirm
◄--	05 64 05 40 00 04 01 00 A3 96	Reset link states
--►	05 64 05 80 01 00 00 04 53 11	Ack
◄--	05 64 53 73 00 04 01 00 03 FC C1 E3 81 96 00 02 01 28 01 00 00 00 01 02 01 28 05 24 01 00 01 00 01 02 01 28 01 00 02 00 01 02 01 28 B4 77 01 00 03 00 01 20 02 28 01 00 00 00 01 00 00 20 A5 25 02 28 01 00 01 00 01 00 00 01 01 01 00 00 03 00 2F AC 00 1E 02 01 00 00 01 00 01 00 00 01 00 00 16 ED	Response. IIN = device restart, need time, class 1 & 2 events. 4 binary input events, 2 analog input events, 4 binary inputs and 2 analog inputs.
--►	05 64 05 80 01 00 00 04 53 11	Link layer confirm
--►	05 64 08 C4 01 00 00 04 A4 CF C1 C3 00 20 3F	Application layer confirm

Reset Restart IIN Bit		
--►	05 64 0E C4 01 00 00 04 7D A4 C0 C4 02 50 01 00 07 07 00 64 11	Request write IIN1.7 = 0
◄--	05 64 0A 44 00 04 01 00 59 5E C2 C4 81 10 00 93 AD	Null response

Set Time and Date		
--►	05 64 12 C4 01 00 00 04 0E 0B C0 C5 02 32 01 07 01 F8 B8 6C AA F0 00 98 98	Request write time and date
◄--	05 64 0A 44 00 04 01 00 59 5E C3 C5 81 00 00 55 93	Null response

Key: --► Master station transmissions (Address 1024 decimal).
◄-- Outstation transmissions (Address 1).

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Annex C

(informative)

Associations

C.1 Introduction

DNP3 over Internet Protocol follows the same logical connection structure as DNP3 over serial ports. The simplest case is a device (either master or outstation) with only one available logical connection to another device. In this case, it is clear that messages arriving on the connection are from the "other" device. The situation becomes more complicated when a device allows simultaneous communication with more than one other device.

For clarity, this annex shall only discuss the issue of solicited messaging and configurations without dual end points.

C.2 Association definition

An association is a representation of a logical connection between a master and an outstation. An association maintains state information associated with the logical connection. This state information includes:

- Master Identifier (e.g., a communication port number if serial or an IP address if networked)
- 16-bit master DNP3 address
- Outstation Identifier
- 16-bit outstation DNP3 address
- Data Link Layer state information
- Transport Function state information
- Application Layer state information
- User Layer state information (e.g., timeout status on a control select)

and for DNP3/UDP or DNP3/TCP associations also includes:

- Remote UDP or TCP port number
- Remote IP address
- Local UDP or TCP port number
- Type of port (either UDP or TCP)

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

DNP3 over serial connections requires a (constant) physical link between devices. In this case, the combination of the physical link and 16-bit DNP3 address uniquely defines the association. In fact, for a point-to-point RS-232 link, the association is completely defined by just the physical link. In the IP network case, a single physical link can "carry" an unlimited number of associations. Each outstation needs to be capable of determining the association identity solely by observing the data "on the wire." It is important to note that associations shall remain intact longer than a single message exchange between a master and an outstation.

C.3 Association issues

Creation of an association using serial ports is a simple matter. From the master perspective, if it wishes to communicate with an outstation, it simply inspects its local pre-configured tables to determine whether it already has established an association with the target device. If not (and if the master has sufficient resources), it allocates resources for the outstation and begins communication using the communication port and outstation DNP3 address from the table. The association with the outstation is never terminated (because the wire between the devices continues to exist). From the outstation perspective, it simply inspects its local tables to determine whether it has ever communicated with this master address. If so, it uses that existing state information. If not, it creates an association that shall initially send the RESTART IIN. This association is never destroyed.

DNP3 transportation over IP-based networks adds additional complications. Whereas there is a fairly limited number of devices that can directly connect via serial ports (1 per serial port for RS-232 and about 32 per serial port for RS-485), there are an unlimited number of devices that may logically connect using the Internet Protocol suite. When using DNP3 over IP, the combination of physical link and DNP3 address is no longer sufficient to uniquely identify the association. In addition, the number of associations that may be maintained by an outstation using DNP3 over IP is limited only by the resources within the outstation and not by the physical connection hardware.

C.4 UDP associations

UDP is a stateless protocol. This means that, at the UDP level, each end of the link has no concept of prior communication. This does not match the DNP3 requirement that each end maintain association state information. Therefore, the master shall maintain information in addition to what is required for UDP communication.

When a master wishes to communicate with an outstation using UDP, it performs the same internal table scan/association creation as for a serial connection. It then constructs an outgoing datagram addressed to the outstation using the IP address, UDP port number, and the identified outstation DNP3 address from its internal table. Incoming datagrams (UDP messages received from the network) are matched to an association using the outstation's IP address and the outstation's UDP port number.

When an outstation receives a request from the master, it shall also match this to an association. Unlike the master case, pre-configured tables need not be present in the outstation; in which case, the outstation shall build the table "on-the-fly." If the outstation determines that this is an existing (active) association, it passes the message to that association's DNP3 protocol layers, which may generate a response. The DNP3 protocol layers use information from the association, in the same way as they would for a serial connection, for example, checking whether the message has the correct application sequence number. The response generated shall also include association-specific information such as whether the RESTART IIN should be set. If the outstation determines that this is a new (permitted) association, it creates the association and sets the application state to indicate that the RESTART IIN should be sent in responses until it is cleared.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

The matching of a UDP message with an association may include the DNP3 destination address, the source IP address, or the source UDP port number. An interesting question is what to do if the incoming UDP datagram is from an IP address with an inactive (timed-out) UDP association. Two choices exist in this case: Ignore the fact that the association was already known and re-create it, or attempt to re-establish communications using the association. The latter is recommended since loss of an association may mean discarding of "good" events. UDP transactions can resume if the network connection is lost and then re-gained. It should be treated the same as if a serial port plug was removed and then re-inserted.

C.5 TCP associations

TCP is a state-oriented protocol. This means that, at the TCP level, each end of the link shall participate in a connection setup before DNP3 communications can commence. When a master wishes to communicate with an outstation using TCP, it performs the same internal table scan/association creation as for a serial connection. It then requests a connection to the outstation and forms the TCP stream, which shall be used for communication. Requests are sent to the outstation using the stream, and responses from the outstation are received on the stream.

When an outstation receives a request from the master, it shall match this to an association. Unlike the master case, pre-configured tables need not be present in the outstation; in which case, the outstation shall build the table on "on-the-fly."

Outstations using DNP3 over TCP receive two types of TCP messages: requests for connection and stream data. Upon a receipt of a TCP request for connection, the outstation performs the same determination of association existence as is done in UDP. The matching of a TCP connection request with an association may include checking the DNP3 destination address, the TCP port number, or the source IP address. If the outstation finds an existing association, it may check the old connection and, if it is no longer in use, re-attach to the existing association. It is recommended that the outstation attempt to re-attach to a matching association, when possible, to avoid discarding of "good" events.

TCP messages with stream data are simply routed to the appropriate DNP3 protocol layers for the logical association, corresponding to the received stream (which may generate responses on that stream). Responses generated by an outstation shall also include association-specific information such as whether the RESTART IIN should be set. If the outstation determines that this is a new (permitted) association, it creates the association and sets the application state to indicate that the RESTART IIN should be sent in responses until it is cleared.

Once connected, each end of the DNP3/TCP link expects periodic messages from the other end. Both devices shall retain an association while messages are received with sufficient regularity (determined by the poll interval or keep-alive timer). If the association was not retained, then, for example, the DNP3 protocol Application Layer would always be forced to accept every sequence number and there would be NO protection against duplicated messages.

After a TCP association times out, the outstation may release the association's resources. A possible implementation may mark a timed-out association as available for use by a new association. In this case, a timeout simply permits release of state information.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Annex D

(normative)

UTF-8 related copyright

When referring to UTF-8 in the XML Schema, IETF RFC 3629 has been used.[43]

IETF RFC 3629 contains the following copyright notice that applies to examples and quotations appearing in this document that were extracted from IETF RFC 3629.

> Copyright (C) The Internet Society (2003). All Rights Reserved.
>
> This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.
>
> The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.
>
> This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

[43] IETF publications are available from the Internet Engineering Task Force, IETF Secretariat, c/o Association Management Solutions, LLC (AMS), 48377 Fremont Boulevard, Suite 117, Fremont, CA 94538 (http://www.ietf.org).

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.

Annex E

(informative)

Bibliography

[B1] DNP User's Group. The Device Profile: Appendix 1 of Volume 8 of the DNP3 Specification. 1st ed. May 2007. Current edition May 2009.

[B2] FIPS 180-2, Secure Hash Standard (includes SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512).[44]

[B3] IEC 60870-5-1:1990, Telecontrol equipment and systems. Part 5: Transmission protocols—Section One: Transmission frame formats.[45]

[B4] IEC 60870-5-2:1992, Telecontrol equipment and systems - Part 5: Transmission protocols—Section 2: Link transmission procedures.

[B5] IEC 60870-5-101:2003, Telecontrol equipment and systems—Part 5-101: Transmission protocols—Companion standard for basic telecontrol tasks.

[B6] IEC/TS 62351-1, Power systems management and associated information exchange—Data and communications security. Part 1: Communication network and system security—Introduction to security issues.

[B7] IEEE Std C37.1™-2007, IEEE Standard for SCADA and Automation Systems.[46]

[B8] IETF RFC 2104, HMAC: Keyed-Hashing for Message Authentication.[47]

[B9] IETF RFC 3174, US Secure Hash Algorithm (SHA-1).

[B10] IETF RFC 3394, Advanced Encryption Standard (AES) Key Wrap Algorithm.

[B11] Internet Assigned Numbers Authority.[48]

[B12] Internet Engineering Task Force.[49]

[B13] ISO 8601, Data elements and interchange formats—Information interchange—Representation of dates and times.

[B14] Stevens, W. R., *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, MA: Addison-Wesley, 1994. (A complete introduction to the Internet Protocols.)

[B15] Mansfield, N., *Practical TCP/IP: Designing, Using, and Troubleshooting TCP/IP Networks on Linux and Windows.* London: Addison-Wesley/Pearson Education Ltd., 2003. (A hands-on tutorial with a wide scope of networking information.)

[44] FIPS publications are available from the National Technical Information Service (NTIS), U. S. Dept. of Commerce, 5285 Port Royal Rd., Springfield, VA 22161 (http://www.ntis.org/).
[45] IEC publications are available from the Sales Department of the International Electrotechnical Commission, Case Postale 131, 3 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (http://www.iec.ch/). IEC publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.
[46] IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).
[47] IETF publications are available from the Internet Engineering Task Force, IETF Secretariat, c/o Association Management Solutions, LLC (AMS), 48377 Fremont Boulevard, Suite 117, Fremont, CA 94538 (http://www.ietf.org).
[48] At the date of the publication, this information was located at http://www.iana.org.
[49] At the date of the publication, this information was located at http://www.ietf.org.

Copyright © 2010 IEEE. All rights reserved.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on July 18,2022 at 15:45:20 UTC from IEEE Xplore. Restrictions apply.