# THE CIP NETWORKS LIBRARY

# Volume 1

# Common Industrial Protocol (CIP™)

Edition 3.20

April 2016


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The CIP Networks Library
Volume 1: Common Industrial Protocol (CIP™)
Publication Number: PUB00001
Copyright © 2001 through 2016 ODVA, Inc. (ODVA). All rights reserved. For permissions to reproduce excerpts of this material, with appropriate attribution to the author(s), please contact ODVA at:

ODVA, Inc.
4220 Varsity Drive, Suite A, Ann Arbor, MI 48108-5006 USA
TEL 1-734-975-8840
FAX 1-734-922-0027
EMAIL odva@odva.org
WEB www.odva.org

Warranty Disclaimer Statement
The right to make, use, or sell product or system implementations based upon the Common Industrial Protocol (CIP) is granted only under separate license pursuant to a Terms of Usage Agreement or other agreement. The ODVA Terms of Usage Agreement is available, at standard charges, over the Internet at www.odva.org. NOTE: Because the technologies described in the CIP Networks Library may be applied in many diverse situations and in conjunction with products and systems from multiple vendors, the user and those responsible for specifying these technologies must determine for themselves their suitability for the intended use. ALL INFORMATION PROVIDED BY ODVA IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE, AND ODVA AND ITS MEMBERS, PARTICIPANTS, SPECIAL INTERESTS GROUPS, EXECUTIVE DIRECTOR AND BOARD OF DIRECTORS EXPRESSLY DISCLAIM ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR OR INTENDED PURPOSE, OR ANY OTHER WARRANTY OTHERWISE ARISING OUT OF THE SPECIFICATIONS. ODVA AND ITS MEMBERS, PARTICIPANTS, SPECIAL INTERESTS GROUPS, EXECUTIVE DIRECTOR AND BOARD OF DIRECTORS DO NOT WARRANT THAT USE OF THE SPECIFICATIONS (INCLUDING, WITHOUT LIMITATION, THE MANUFACTURE, DISTRIBUTION AND SALE OF PRODUCTS THAT COMPLY WITH THE SPECIFICATIONS) WILL BE ROYALTY-FREE. The user should always verify interconnection requirements to and from other equipment, and confirm installation and maintenance requirements for their specific application. IN NO EVENT SHALL ODVA, ITS OFFICERS, DIRECTORS, MEMBERS, AGENTS, LICENSORS, OR AFFILIATES BE LIABLE TO YOU, ANY CUSTOMER, OR THIRD PARTY FOR ANY DAMAGES, DIRECT OR INDIRECT, INCLUDING BUT NOT LIMITED TO LOST PROFITS, DEVELOPMENT EXPENSES, OR ANY OTHER DIRECT, INDIRECT, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES.

The following are trademarks of ODVA:
CIP, CIP Energy, CIP Motion, CIP Security, CIP Safety, CIP Sync, CompoNet, ControlNet, DeviceNet, EtherNet/IP, ODVA CONFORMANT, QuickConnect.
All other trademarks referenced herein are property of their respective owners.

SITE SUBSCRIPTION

- The Final Specification, of which this volume and edition of The CIP Networks Library is a part, is provided on an annual subscription basis to this Licensed Vendor Member, as defined by its unique Vendor ID for the technology contained in the Final Specification ("YOU"), pursuant to your Terms of Usage Agreement with ODVA, Inc. (ODVA) for the technology contained in the Final Specification.
- This subscription is a site subscription, and this subscription, along with your membership in ODVA, must be renewed annually in order to maintain continuing rights to the site subscription as allowed under your Terms of Usage Agreement.
- This site subscription permits access to the electronic files for this volume contained on the distribution CD by multiple users who are your employees and on-site contracted individuals performing typical employee functions on a contract basis ("Authorized Users"). YOU shall ensure that, if access to these files if given to contractors, the contractors can fulfill the obligations of your Terms of Usage Agreement for the ODVA technology contained in the Final Specification. YOU shall not knowingly permit anyone other than Authorized Users to access these files.
- This site subscription permits access to the electronic files contained on the distribution CD for this volume via the original CD on which this volume is distributed or via an electronic copy of this volume placed on your single, secure intranet site. If and when the electronic files are posted on your intranet site, YOU shall relocate the original CD to secure storage for use only as a system back-up for the electronic files posted on your intranet site.
- The possession of or subscription to this Final Specification does not, by itself, convey any right to use or reproduce any portion of the Final Specification or to make, have made, use, import, offer to sell, sell, lease, market, or otherwise distribute or dispose of any products contemplated by the Final Specification, and you are hereby notified that the products contemplated by this Final Specification might be covered by valid patents or copyrights of ODVA, its members or other licensors. The necessary licenses to use or reproduce portions of the Final Specification for use in products, or to make, have made, use, import, offer to sell, sell, lease, market, or otherwise distribute and dispose of such products may be obtained only from ODVA through its Terms of Usage Agreement, available through the ODVA web site. This license requirement applies equally (a) to devices that completely implement the Final Specification with a network port that can be issued a Declaration of Conformity ("CIP Network Devices"), (b) to components of such CIP Network Devices to the extent they implement portions of the Final Specification, and (c) to enabling technology products, such as any CIP Network protocol stack, designed for use in CIP Network Devices to the extent they implement portions of the Final Specification. Contact ODVA for a Terms of Usage Agreement if you are not already licensed.
- Any other distribution and all other electronic copies, intranet or internet postings, and any printed copies are prohibited. Printed copies of this volume may be purchased via the order form available through the ODVA web site at www.odva.org.
- Notwithstanding anything to the contrary herein, if in connection with bookmarking a page of the Final Specification it is reasonably necessary for an Authorized User to possess a copy of the Final Specification on the computer on which such page is bookmarked, then you may possess such copy, provided that (i) such copy is not accessible to anyone other than the Authorized User, (ii) such copy is not retained for any longer than reasonably necessary to use the bookmarked page and in any event no longer than the term of this subscription, (iii) use of such copy is limited to the uses permitted in this site subscription and (iv) such copy is not transmitted or otherwise distributed to any other person, computer or device.

This page is intentionally left blank


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# The CIP Networks Library: Volume 1

# Common Industrial Protocol (CIP™)

## Table of Contents


| | |
|---|---|
| **Revisions** | - Summary of Changes in this Edition |
| **Preface** | - Organization of CIP Networks Specifications |
| | - The Specification Enhancement Process |
| **Chapter 1** | - Introduction to the Common Industrial Protocol |
| **Chapter 2** | - Messaging Protocol |
| **Chapter 3** | - Communications Objects |
| **Chapter 4** | - CIP Object Model |
| **Chapter 5A** | - Object Library, Part A |
| **Chapter 5B** | - Object Library, Part B |
| **Chapter 6** | - Device Profiles |
| **Chapter 7** | - Electronic Data Sheets |
| **Chapter 8** | - Physical Layer |
| **Chapter 9** | - Indicators and Middle Layers |
| **Chapter 10** | - Bridging and Routing |
| **Appendix A** | - Explicit Messaging Services |
| **Appendix B** | - Status Codes |
| **Appendix C** | - Data Management |
| **Appendix D** | - Engineering Units |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Revisions

This edition of the CIP Networks Library Volume 1: Common Industrial Protocol contains the following changes from the previously released edition. Please see the change bars on the pages noted here for specific modifications. Note: Some of the pages within the ranges noted may not contain any changes.

| Chapter | Pages | Description of Change |
|---|---|---|
| | | **Additions for Big 12 Diagnostics** |
| 3-5.2 | 3-50 | • Change semantics cell for attribute 11 to refer to new section<br>• Add attributes 15, 16, 17. 18, 19 & 20 |
| 3-5.2.1.1<br>3-5.2.2.3<br>3-5.2.2.4 | 3-50, 51 | • Insert new sections for Attribute 11, 15 & 16 semantics |
| 3-5.2.2.5<br>thru<br>3-5.2.2.8 | 3-52 | • Insert new sections for Attributes 17, 18, 19 & 20 semantics |
| | | **Originator Connection List Object Clarification** |
| 5B-4 | 5B-43 | • Add new last paragraph |
| 5B-4.3.2 | 5B-45 | • Changes to 2nd and 3rd paragraphs |
| 5B-4.3.3 | 5B-46 | • Change “Structure_size” to “Size” and edit description in Table 5B-5.9 |
| 5B-11 | 5B-164 | • Add new 2nd paragraph |
| | | **Communications Objects Modelling** |
| 1-7 | 1-16 | • Add definition 1-7.12 CIP Routable Port |
| 3-7 | 3-99 | • Changes to last paragraph |
| 3-7.2 | 3-102 | • Add new Attribute 11 |
| 3-7.2.1.9 | 3-106 | • Add new section for Attribute 11 semantics |
| 5A-1 | 5A-14 to 16 | • Add new column to table and footnote |
| | 5A-17 to 19 | • Add new table with different sort |
| 5B-11 | 5B-164 | • Remove NOTE that discussed scope of objects |
| 7-3.5.1.15 | 7-27 | • Add new section for Array of EPATHs format convention |
| 7-3.6.12 | 7-105 | • Changes to Table 7-3.60 |
| 10-3 | 10-3 | • Replaced all of section with new paragraph |
| C-1.4.1 | C-6 | • Removed mention of port addressable objects from 6th paragraph |
| | | **Support for Arrays in EDS** |
| 7-3.5.3.7 | 7-30 | • Add new entry keywords to list, change to last paragraph of section |
| 7-3.6.5.4 | 7-55 | • Add ArrayN to CfgAssemblyExa Keyword Format table |
| 7-3.6.8 | 7-66 | • Add new entry names to entry keywords table |
| 7-3.6.8.1 | 7-67 | • Add ArrayN to Member Reference Data Type in AssemExaN Entry Format table |
| 7-3.6.8.1.4 | 7-68 | • Change in 2nd paragraph and table |
| 7-3.6.8.2 | 7-74, 75 | • Add ArrayN to Data Type field in several tables |
| 7-3.8.6.3 | 7-83 thru<br>7-86 | • Insert new section and subsections to describe Array Entry Keyword |
| 7-3.6.10.1 | 7-95 | • Add ArrayN to Data Type of fields 5, 8, 10 & 12 in table |
| 7-3.6.16 | 7-120 | • Add ArrayN to first paragraph and table |
| 7-3.6.17 | 7-123 | • Add ArrayN to EDS Entry Keywords in Int. Section table |
| 7-3.6.17.10 | 7-127 | • Insert new section/subsections for Array Entry Keywords |
| C-2.1.1 | C-22 | • Corrected capitalization of UINT in footnote 14 of table |
| C-4.1 | C-35 | • Changed “octet string” to UINT in ENGUNIT |
| C-4.2 | C-36 | • Change to FormalArrayTypeSpec |
| C-5.2.8 | C-42, 43 | • Remove references to ASN.1, changes to first paragraph and Array definitions, change example ARRAY1 |
| C-5.2.10 | C-44 | • Change Example 1 ARRAY1 and Example 2 ARRAY |
| C-6.2.2 | C-50 to 52 | • Changes to Formal Array Encoding Definition table |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Chapter | Pages | Description of Change |
|---|---|---|
| | | **Changes due to the creation of Vol 9 and misc changes** |
| 5B-4.3.3 | 5B-46 | • Changed references of O→T and T→O for consistency in Table 5B-4.9 |
| 5B-5.3.1 | 5B-50 | • Added missing "O" to T→O reference in first paragraph |
| 6-41.4 | 6-244 | • Figure 6-41.1 was poorly rendered, redrew |
| 5A-1 | 5A-14 | • Changed reference for Motion Device Axis object to Volume 9 |
| 5B-2 | - | • Move CIP Motion object from Chapter 5B to Volume 9 (changes to subsequent headings, tables and figures are not marked) |
| 6-7 | 6-19 to 21 | • Changed CIP Motion Device Profile references to Volume 9 |
| 6-41 | - | • Move CIP Motion Device Profile from Chapter 6 to Volume 9 (changes to subsequent headings, tables and figures are not marked) |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Preface

Organization of the CIP Networks Specifications

Today, four networks - DeviceNet™, ControlNet™, EtherNet/IP™ and CompoNet™ - use the Common Industrial Protocol (CIP) for the upper layers of their network protocol. For this reason, ODVA manages and distributes the specifications for CIP Networks in a common structure to help ensure consistency and accuracy in the management of these specifications.

The following diagram illustrates the organization of the library of CIP Network specifications. In addition to CIP Networks, CIP Safety™ consists of the extensions to CIP for functional safety.

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This common structure presents CIP in one volume with a separate volume for each network adaptation of CIP. The specifications for the CIP Networks are two-volume sets, paired as shown below.

The EtherNet/IP Specification consists of:

Volume 1: Common Industrial Protocol (CIP™)
Volume 2: EtherNet/IP Adaptation of CIP

The DeviceNet Specification consists of:

Volume 1: Common Industrial Protocol (CIP™)
Volume 3: DeviceNet Adaptation of CIP

The ControlNet Specification consists of:

Volume 1: Common Industrial Protocol (CIP™)
Volume 4: ControlNet Adaptation of CIP

The CompoNet Specification consists of:

Volume 1: Common Industrial Protocol (CIP™)
Volume 6: CompoNet Adaptation of CIP

The Specification for CIP Safety™ is distributed in a single volume:

Volume 5: CIP Safety™

The Specification for integrating Modbus Devices is distributed in a single volume:

Volume 7: Integration of Modbus Devices into the CIP Architecture

The Specification for CIP Security is distributed in a single volume:

Volume 8: CIP Security™

The specification for CIP Motion is distributed in a single volume:

Volume 9: CIP Motion™

Specification Enhancement Process

The specifications for CIP Networks are continually being enhanced to meet the increasing needs of users for features and functionality. ODVA has implemented a Specification Enhancement Process in order to ensure open and stable specifications for all CIP Networks. This process is ongoing throughout the year for each CIP Network Specification as shown in the figure below. New editions of each CIP Network specification are published on a periodic basis.

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This page is intentionally left blank


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Volume 1: Common Industrial Protocol (CIP™) Specification

# Chapter 1: Introduction to CIP


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Contents

1-1 Introduction ..... 3
1-2 Object Modeling ..... 4
1-2.1 Object Addressing ..... 6
1-3 Network Overview ..... 9
1-3.1 I/O Connections ..... 10
1-3.2 Explicit Messaging Connections ..... 11
1-4 CIP Object Model ..... 12
1-5 System Structure ..... 13
1-5.1 Topology ..... 13
1-5.2 Logical Structure ..... 14
1-6 CIP Specification Structure ..... 15
1-7 Definitions ..... 16
1-8 Abbreviations ..... 19


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 1-1 Introduction

The Common Industrial Protocol (CIP) is a peer to peer object oriented protocol that provides connections between industrial devices (sensors, actuators) and higher-level devices (controllers). CIP is physical media and data link layer independent. See Figure 1-1.1.

**Figure 1-1.1 Example CIP Communication Link**

CIP has two primary purposes:

- Transport of control-oriented data associated with I/O devices
- Transport of other information that is related to the system being controlled, such as configuration parameters and diagnostics.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 1-2 Object Modeling

CIP makes use of abstract *object modeling* to describe:

- The suite of communication services available
- The externally visible behavior of a CIP node
- A common means by which information within CIP products is accessed and exchanged

A CIP node is modeled as a collection of *Objects*. An ***Object*** provides an abstract representation of a particular component within a product. The realization of this abstract object model within a product is implementation dependent. In other words, a product internally maps this object model in a fashion specific to its implementation.

A ***Class*** is a set of Objects that all represent the same kind of system component. An ***Object Instance*** is the actual representation of a particular Object within a Class. Each *Instance* of a Class has the same set of attribute definitions. Each *Instance* need only support the optional attributes that apply to it. Each *Instance* has its own set of attribute values. As Figure 1-2.1 illustrates, multiple *Object Instances* within a particular Class can reside in a CIP node.

**Figure 1-2.1 A Class of Objects**

An Object Instance and/or an Object Class has *Attributes*, provides *Services*, and implements a *Behavior*.

***Attributes*** are characteristics of an Object and/or an Object Class. Typically, Attributes provide status information or govern the operation of an Object. ***Services*** are invoked to trigger the Object/Class to perform a task. The ***Behavior*** of an Object indicates how it responds to particular events. For example, a person can be abstractly viewed as an Instance within the Class *Human*. Generally speaking, all humans have the same set of attributes: age, gender, etc., yet, because the values of each attribute vary, each of us looks/behaves in a distinct fashion.

**Table 1-2.1 Object Model Terminology Examples**

| Class | Instances | Attributes | Attribute Values |
|---|---|---|---|
| Human | Mary | Gender | Female |
| | | Age | 31 |
| | Jerry | Gender | Male |
| | | Age | 50 |

The following Object Modeling related terms are used when describing CIP services and protocol.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- **Object –** An abstract representation of a particular component within a product.
- **Class –** A set of objects that all represent the same kind of system component. A class is a generalization of an object. All objects in a class are identical in form and behavior, but may contain different attribute values.
- **Instance –** A specific and real (physical) occurrence of an object. For example: New Zealand is an instance of the object class Country. The terms Object, Instance, and Object Instance all refer to a specific Instance.
- **Attribute –** A description of an externally visible characteristic or feature of an object. Typically, attributes provide status information or govern the operation of an Object. For example: the ASCII name of an object; and the repetition rate of a cyclic object.
- **Instantiate -** To create an instance of an object with all instance attributes initialized to zero unless default values are specified in the object definition.
- **Behavior** – A specification of how an object acts. Actions result from different events the object detects, such as receiving service requests, detecting internal faults or elapsing timers.
- **Service** – A function supported by an object and/or object class. CIP defines a set of common services and provides for the definition of Object Class and/or Vendor Specific services. CIP common services are those whose parameters and required behaviors are defined in Appendix A.
- **Communication Objects -** A reference to the Object Classes that manage and provide the run-time exchange of implicit (I/O) and explicit messages.
- **Application Objects -** A reference to multiple Object Classes that implement product-specific features.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 1-2.1 Object Addressing

The information in this section provides a common basis for **logically** addressing separate physical components across CIP. The following list describes the information that is used to address an Object from a CIP network:

- **Media Access Control Identifier (MAC ID) -** An identification value assigned to each node on the CIP network. This value distinguishes a node among all other nodes on the same link and is typically used as the node address. However on EtherNet/IP the IP Address is used for the node address and not the Ethernet MAC ID. In this volume, references made to MAC ID are to be interpreted as the Node Address. Each network specific volume defines what address identifier is to be used as the Node Address.

**Figure 1-2.2 MAC ID Example**

- **Class Identifier (Class ID) -** An integer identification value assigned to each Object Class accessible from the network.

**Figure 1-2.3 Class ID Example**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- **Instance Identifier (Instance ID) -** An integer identification value assigned to an Object Instance that identifies it among all Instances of the same Class. This integer is unique within the MAC ID:Class in which it resides.

**Figure 1-2.4 Instance Identifier Example**

It is also possible to address the Class itself versus a specific Object Instance within the Class. This is accomplished by utilizing the Instance ID value zero (0). **CIP reserves the Instance ID value zero (0) to indicate a reference to the Class versus a specific Instance within the Class.**

**Figure 1-2.5 Instance #0 Example**

- **Attribute Identifier (Attribute ID) -** An integer identification value assigned to a Class and/or Instance Attribute.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 1-2.6 Attribute ID Example**

- **Service Code** - An integer identification value which denotes a particular Object Instance and/or Object Class function.

**Figure 1-2.7 Service Code Example**

CIP defined ranges for the Object Addressing information presented above are referred to using the following terms:

- **Open -** A range of values whose meaning is defined by ODVA. These CIP-defined values provide commonality and consistency amongst all CIP implementations. Chapter 4-9.1 defines the open ranges for Class IDs, Attribute IDs and Service Codes.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- **Vendor Specific -** A range of values specific to the vendor of a device. These are used by vendors to extend their devices beyond the available Open options. A vendor internally manages the use of values within this range. Chapter 4-9.2 discusses extending objects, including adding vendor specific extensions.
- **Object Class Specific -** A range of values whose meaning is defined by an Object Class. This range applies to Service Code definitions. The object definitions found throughout the volumes of the CIP Networks Library define object class specific services for each class.

## 1-3 Network Overview

CIP defines a connection-based scheme to facilitate all application communications. A CIP ***connection*** provides a communication path between multiple end-points. The end-points of a connection are applications that need to share data. Transmissions associated with a particular connection are assigned an identification value when a connection is established. This identification value is called the ***Connection ID (CID)***.

***Connection Objects*** model the communication characteristics of a particular Application-to-Application(s) relationship. The term ***end-point*** refers to one of the communicating entities involved in a connection.

CIP's connection-based scheme defines a dynamic means by which the following two types of connections can be established:

- **I/O Connections -** Provide dedicated, special-purpose communication paths between a producing application and one or more consuming applications for the purpose of moving application-specific data. This is often referred to as implicit messaging
- **Explicit Messaging Connections -** Provide generic, multi-purpose communication paths between two devices. These connections often are referred to as just Messaging Connections. Explicit Messages provide the typical request/response-oriented network communications.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 1-3.1 I/O Connections

As previously stated, I/O *Connections* provide special-purpose communication paths between a producing application and one or more consuming applications. Application-specific I/O data moves across an I/O Connection.

***I/O Messages*** are exchanged across I/O connections. An I/O Message consists of a Connection ID and optional associated I/O data. The meaning of the data within an I/O Message is *implied* by the associated Connection ID. The connection end-points are assumed to have knowledge of the intended use or meaning of the I/O Message.

**Figure 1-3.1 CIP I/O Connection**

**This document does not define any particular use for I/O Messaging**. There are a wide variety of functions that can be accomplished using I/O Messaging. Either by virtue of the particular type of product transmitting an I/O Message, or based upon configuration performed using Explicit Messaging, the meaning and/or intended use of all I/O Messages can be made known to the system. See Chapter 6 Device Profiles for I/O message formats defined for common device types.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 1-3.2 Explicit Messaging Connections

Explicit Messaging Connections provide generic, multi-purpose communication paths between two devices. Explicit Messages are exchanged across Explicit Messaging Connections. **Explicit Messages** are used to command the performance of a particular task and to report the results of performing the task. Explicit Messaging provides the means by which typical request/response oriented functions are performed (e.g. module configuration).

CIP defines an Explicit Messaging protocol that states the meaning of the message. An Explicit Message consists of a Connection ID and associated messaging protocol information.

**Figure 1-3.2 CIP Explicit Messaging Connection**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 1-4 CIP Object Model

Figure 1-4.1 illustrates the abstract object model of a CIP product. Included are the following components:

- **Unconnected Message Manager (UCMM) -** Processes CIP Unconnected Explicit messages.
- **Connection Object Class-** Allocates and manages internal resources associated with both I/O and Explicit Messaging connections.
- **Connection Object Instance-** Manages the communication-specific aspects associated with a particular application-to-application network relationship.
- **Connection Manager –** An object class, required in some CIP network nodes, used to control certain aspects of Connection Object instances within the node.
- **Network-Specific Link Object -** Provides the configuration and status of a physical CIP network connection (eg. DeviceNet and ControlNet objects).
- **Message Router -** Distributes Explicit Request Messages to the appropriate handler object.
- **Application Objects -** Implement the intended purpose of the product.

**Figure 1-4.1 CIP Device Object Model**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 1-5 System Structure

## 1-5.1 Topology

The system structure uses the following physical organization.

- **System = { Domain(s) }**
  - System contains one or more domains.
- **Domain = { Network(s) }**
  - A domain is a collection of one or more networks. Networks must be unique within a domain. A domain may contain a variety of network types.
- **Network = { Subnet(s) }**
  - A network is a collection of one or more subnets, where each node's MAC ID is unique on the network.
- **Subnet = { Nodes(s) }**
  - A subnet is a collection of nodes using a common protocol and shared media access arbitration. i.e. subnet may have multiple physical segments and contain repeaters.
- **Segment = { Nodes(s) }**
  - A segment is a collection of nodes connected to a single uninterrupted section of physical media.
- **Node = { Object(s) }**
  - A node is a collection of objects that communicate over a subnet, and arbitrates using a single MAC ID. A physical device may contain one or more nodes.

**Figure 1-5.1 System Structure - Topology**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- **Rp** Segment Repeater between segments
  - Segments participate in the same media arbitration
- B B Subnet Bridge between subnets
  - Duplicate MAC ID check passes through
  - MAC ID's on one subnet shall not be duplicated on the other subnet
  - Subnets may operate at different baud rates
- R R Network Router between similar networks
  - Both networks are CIP-based
- G G Gateway between dissimilar networks
  - One network is CIP-based, the other is not

## 1-5.2 Logical Structure

The system structure uses the following logical elements.

- Cluster = { Node(s) }
  - A cluster is a collection of nodes that are logically connected. A node may belong to one or more clusters. A cluster may span subnets, networks or domains.

**Figure 1-5.2 Clusters**

**Cluster A -** Master/Slave Point-to-point Communication (i.e. Poll/Cyclic/COS)

- A Master and it's Slaves are a cluster
- A Master may also be a slave to another Master

**Cluster B -** Multicast Master/Slave Communication (i.e. Strobe)

- A Master and its Slaves are a cluster
- A Master may also participate with a peer in any cluster

**Cluster C -** Peer-to-peer Communication (point-to-point or multicast)

- Nodes participating in a particular peer-to-peer relationship are a cluster


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 1-6 CIP Specification Structure

The CIP Networks Library is the definition of the application and user layers for a number of different network adaptations. The figure below shows the relationship between the specifications of this document and those of the CIP-based networks.

**Figure 1-6.1 CIP Architecture and Related Specifications**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 1-7 Definitions

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

| Term | | Definition |
|---|---|---|
| 1-7.1 | actual packet interval (API) | The measure of how frequently a specific connection produces its data. |
| 1-7.2 | allocate | To take a resource from a common area and assign that resource for the exclusive use of a specific entity. |
| 1-7.3 | application | Function or data structure for which data is consumed or produced. |
| 1-7.4 | application objects | Multiple object classes that manage and provide the run-time exchange of messages across the network and within the network device. |
| 1-7.5 | attribute | A description of an externally visible characteristic or feature of an object. The attributes of an object contain information about variable portions of an object. Typically, they provide status information or govern the operation of an object. Attributes may also affect the behavior of an object. Attributes are divided into class attributes and instance attributes. |
| 1-7.6 | backplane | An internal bus that connects modules in a rack/chassis |
| 1-7.7 | behavior | Indication of how the object responds to particular events. Its description includes the relationship between attribute values and services. |
| 1-7.8 | big endian | A format for storage or transmission of binary data in which the most significant bit (or byte) comes first. The term comes from "Gulliver's Travels" by Jonathan Swift. The Lilliputians, being very small, had correspondingly small political problems. The Big-Endian and Little-Endian parties debated over whether soft-boiled eggs should be opened at the big end or the little end. See also: little-endian. [Source: RFC1392] |
| 1-7.9 | bit | A unit of information consisting of a 1 or a 0. This is the smallest data unit that can be transmitted. |
| 1-7.10 | byte | See octet. |
| 1-7.11 | chassis | A container of modules connected by a backplane. Synonymous with rack. |
| 1-7.12 | CIP Routable Port | A port that can receive CIP messages and forward them to another (or back out the same) port. |
| 1-7.13 | class | A set of objects all of which represent a similar system component. A class is a generalization of the object, a template for defining variables and methods. All objects in a class are identical in form and behavior, but they may contain different attribute values. |
| 1-7.14 | class specific service | A service defined by a particular object class to perform a required function that is not performed by a common service. A class specific object is unique to the object class that defines it. |
| 1-7.15 | client | (1) An object that uses the services of another (server) object to perform a task.<br>(2) An initiator of a message to which a server reacts. |
| 1-7.16 | communication objects | Components that manage and provide run-time exchange of messages across the network such as the Connection Manager object, the unconnected message manager (UCMM), and the Message Router object. |
| 1-7.17 | connection | A logical binding between two application objects. These application objects may be the same or different devices. |
| 1-7.18 | connection ID (CID) | Identifier assigned to a transmission that is associated with a particular connection between producers and consumers that identifies a specific piece of application information. |
| 1-7.19 | connection path | Is made up of a byte stream that defines the application object to which a connection instance applies. |
| 1-7.20 | consume | The act of receiving data from a producer. |
| 1-7.21 | consumer | A node that is receiving data from a producer. |
| 1-7.22 | consuming application | The application that consumes data. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Term | Definition |
|---|---|
| 1-7.23 cyclic | Term used to describe events that repeat in a regular and repetitive manner. |
| 1-7.24 device | A physical hardware connection to the link. A device may contain more than one node. |
| 1-7.25 device profile | A collection of device-dependent information and functionality providing consistency between similar devices of the same device type. |
| 1-7.26 end node | A producing or consuming node. |
| 1-7.27 end point | One of the communicating entities involved in a connection. |
| 1-7.28 error | A discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition. |
| 1-7.29 instance | The actual physical presentation of an object within a class. Identifies one of many objects within the same object class. |
| 1-7.30 instantiated | An object that has been created in a device. |
| 1-7.31 library element | A derived or standard data type, function, function block, program or resource in IEC 1131 - programmable controllers. |
| 1-7.32 link | Collection of nodes with unique MAC IDs. Segments connected by repeaters make up a link; links connected by routers make up a network. |
| 1-7.33 little endian | Describes a model of memory organization that stores the least significant byte at the lowest address. On the network medium, the lowest order byte is transferred first. The native CIP data types are sent in little endian order. See Appendix C for a detailed description of the specific encoding of each data type. |
| 1-7.34 Message Router | The object within a node that distributes messaging requests to the appropriate application objects. |
| 1-7.35 multicast | A packet with a special destination address, which multiple nodes on the network may be willing to receive. [Source: RFC1392] |
| 1-7.36 multicast connection | A connection from one node to many. Multicast connections allow a single producer to be received by many consumer nodes. |
| 1-7.37 network | A series of nodes connected by some type of communication medium. The connection paths between any pair of nodes can include repeaters, routers and gateways. |
| 1-7.38 network address or node address | An identification value assigned to each node on the CIP network. This value distinguishes a node among all other nodes on the same link. The format is network specific. |
| 1-7.39 new data | Term used to describe data provided by the application versus data used to keep a connection active. |
| 1-7.40 node | A connection to a link that requires a single MAC ID. |
| 1-7.41 object | (1) An abstract representation of a particular component within a product. Objects can be composed of any or all of the following components:<br>a) data (information which changes with time);<br>b) configuration (parameters for behavior);<br>c) methods (things that can be done using data and configuration).<br>(2) A collection of related data (in the form of variables) and methods (procedures) for operating on that data that have clearly defined interface and behavior. |
| 1-7.42 object specific service | A service defined by a particular object class to perform a required function that is not performed by a common service. An object specific service is unique to the object class that defines it. |
| 1-7.43 octet | An octet is 8 bits that indicates no particular data type. |
| 1-7.44 ODVA, Inc. | The organization that supports network technologies built on the Common Industrial Protocol (CIP ™) – DeviceNet, EtherNet/IP, ControlNet and CompoNet. |
| 1-7.45 originator | The client responsible for establishing a connection path to the target. |
| 1-7.46 point-to-point connection | A connection that exists between two nodes only. Connections can be either point-to-point or multicast. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Term | Definition |
|---|---|
| 1-7.47 port | A CIP port is the abstraction for a physical network connection to a CIP device. A CIP device has one port for each network connection. Note: network specific definitions may include additional definitions of this term within the context of the network. |
| 1-7.48 produce | Act of sending data to a consumer. |
| 1-7.49 producer | A node that is responsible for transmitting data. |
| 1-7.50 rack | A container of modules connected by a backplane. Synonymous with chassis. |
| 1-7.51 redundant media | A system using more than one medium to help prevent communication failures. |
| 1-7.52 requested packet interval (RPI) | The measure of how frequently the originating application requires the transmission of data from the target application. |
| 1-7.53 router | A device that can receive CIP messages on one or more CIP ports and forward them to another (or back out the same port). The term router when used by itself refers to a CIP router. |
| 1-7.54 scheduled network | A network which utilizes a time-based scheduling mechanism that provides network devices with deterministic and predictable access to the medium while preventing network collisions. |
| 1-7.55 serial number | A unique 32-bit integer assigned by each manufacturer to every device. The number need only be unique with respect to the manufacturer. |
| 1-7.56 server | An object that provides services to another (client) object. |
| 1-7.57 service | Operation or function that an object performs upon request from another object. |
| 1-7.58 target | The end-node to which a connection is established. |
| 1-7.59 tool | An executable software program that interacts with the user to perform some function. |
| 1-7.60 unconnected message manager (UCMM) | The component within a node that transmits and receives unconnected explicit messages and sends them directly to the Message Router object. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 1-8 Abbreviations

For the purposes of this standard, the following abbreviations apply.

| Abbreviation | Meaning |
|---|---|
| 1-8.1 API | Actual Packet Interval |
| 1-8.2 ASCII | American Standard Code for Information Interchange |
| 1-8.3 ASN.1 | Abstract Syntax Notation |
| 1-8.4 CIP | The Common Industrial Protocol defined in this volume of the CIP Networks Library. CIP includes both connected and unconnected messaging. |
| 1-8.5 CID | Connection Identifier |
| 1-8.6 DLL | Data Link Layer |
| 1-8.7 EPR | Expected Packet Rate |
| 1-8.8 ISO | International Standards Organization |
| 1-8.9 MAC ID | Media Access Control Identifier |
| 1-8.10 PDU | Protocol Data Unit |
| 1-8.11 ODVA | ODVA, Inc. (see Definitions, 1-7.40) |
| 1-8.12 O→T | Originator to Target (used to describe packets that are sent from the originator to the target) |
| 1-8.13 OSI | Open Systems Interconnection (see ISO 7498) |
| 1-8.14 RPI | Requested Packet Interval |
| 1-8.15 SDU | Service Data Unit |
| 1-8.16 SEM | State Event Matrix |
| 1-8.17 SEMI | Semiconductor Equipment Materials International |
| 1-8.18 STD | State Transition Diagram, used to describe object behavior |
| 1-8.19 T→O | Target to Originator (used to describe packets that are sent from the target to the originator) |
| 1-8.20 UCMM | Unconnected Message Manager |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This page is intentionally left blank


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Volume 1: Common Industrial Protocol (CIP™) Specification

# Chapter 2: Messaging Protocol


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Contents

2-1 Introduction ..... 3
2-2 Connection Establishment Overview ..... 4
2-2.1 Explicit Messaging and the UCMM ..... 4
2-2.2 I/O Connections ..... 7
2-3 Client and Server Connection Endpoints ..... 9
2-4 Message Router Request/Response Formats ..... 9
2-4.1 Message Router Request ..... 9
2-4.1.1 Request_Path ..... 9
2-4.2 Message Router Response ..... 10


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 2-1 Introduction

CIP is layered on top of a *connection*-based network. A CIP *connection* provides a path between multiple applications. When a connection is established, the transmissions associated with that connection are assigned a ***Connection ID (CID)***. If the connection involves a bi-directional exchange, then two Connection ID values are assigned. See Figure 2-1.1.

**Figure 2-1.1 Connections and Connection IDs**

The definition and format of the connection ID is network dependent. For example, the connection ID for CIP connections over DeviceNet is based on the CAN Identifier Field.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 2-2 Connection Establishment Overview

This section presents an overview of dynamically establishing both Explicit Messaging and I/O Connections.

### 2-2.1 Explicit Messaging and the UCMM

The Unconnected Message Manager (UCMM) is responsible for processing Unconnected Explicit Requests and Responses. This includes establishing both Explicit Messaging and I/O connections. The underlying network defines how the UCMM is accessed and may limit the messaging which can occur across the UCMM.

When using the UCMM to establish an explicit messaging connection, the target application object is the Message Router object (Class Code 2).

Figure 2-2.1 illustrates the steps involved in establishing a Messaging Connection.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 2-2.1 Establishing an Explicit Messaging Connection**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Explicit Messaging Connections are unconditionally ***point-to-point***. Point-to-point connections exist between two devices ONLY. The device that requests that the connection be opened (the client) is one *end-point* of the connection, and the module that receives and responds to the request (the server) is the other *end-point*. See Figure 2-2.2.

**Figure 2-2.2 Point-to-Point Nature of Explicit Messaging Connections**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 2-2.2 I/O Connections

The CIP Network definition allows the establishment of a variety of I/O connections. This specification does not dictate any rules associated with *who* may perform Connection configuration. For example, a tool could interface with two separate devices and create an I/O Connection between them. See Figure 2-2.3.

**Figure 2-2.3 Tool Interface with Devices to Create Connection**

The tool uses various Explicit Messaging Services to create and configure the I/O Connection Objects within the end points.

I/O connections can be either point-to-point or ***multicast***. Multicast connections allow a single transmission to be heard by many nodes. See Figure 2-2.4.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 2-2.4 Point-to-Point or Multicast Nature of I/O Connections**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 2-3 Client and Server Connection Endpoints

The terms ***Client*** and ***Server*** are used throughout this document when discussing the behavior associated with a connection end-point. A Client end-point and Server end-point(s) are associated with both Explicit Messaging and I/O Connections. The *Client* is the module that originates a transmission, and the *Server* is the module that reacts to that transmission. The Server's reaction may cause it to return a message to the Client.

**Figure 2-3.1 Illustration of Client/Server Message Flow**

## 2-4 Message Router Request/Response Formats

CIP defines a standard data format for delivering data to and from the Message Router object (see Chapter 5A-3). This data format is used in various places within CIP including the Unconnected Send service of the Connection Manager object and the UCMM data structures of most of the CIP networks.

### 2-4.1 Message Router Request

The Message Router Request Format is defined in Table 2-4.1.

**Table 2-4.1 Message Router Request Format**

| Parameter Name | Data Type | Description |
|---|---|---|
| Service | USINT | Service code of the request. |
| Request_Path_Size | USINT | The number of 16 bit words in the Request_Path field (next element). |
| Request_Path | Padded EPATH | This is an array of bytes whose contents convey the application path and other information of the request for this transaction. See section 2-4.1.1. |
| Request_Data | Array of octet | Service specific data per object definition to be delivered in the Explicit Messaging Request. If no additional data is to be sent with the Explicit Messaging Request, then this array will be empty. |

#### 2-4.1.1 Request_Path

The request path shall be of the form:

[Electronic Key segment]
Application Path


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The Electronic Key segment is conditional. If the request is being sent across a connection the Electronic Key segment is not allowed, otherwise it is optional. If the Electronic Key segment is present the target shall evaluate the key. See section C-1.4.2 for the definition of Electronic Key segment and section C-1.5 for the definition of the Application Path format.

Request Path example:

| | |
|---|---|
| 34 04 42 42 0C 00 01 00 81 01 | Electronic Key segment indicating the target device must be compatible with a Vendor 0x4242, Device Type 0x000C, Product Code 0x0001, Major Revision 0x01, Minor Revision 0x01 device |
| 20 04 24 01 30 03 | Application path that specifies Instance 1 of the Assembly Object, Attribute 3. |

## 2-4.2 Message Router Response

The Message Router Response Format is defined in Table 2-4.2.

**Table 2-4.2 Message Router Response Format**

| Parameter Name | Data Type | Description |
|---|---|---|
| Reply Service | USINT | Reply service code is the request service code plus 0x80. |
| Reserved | octet | Shall be zero. |
| General Status | USINT | One of the General Status codes listed in Appendix B (Status Codes). |
| Size of Additional Status | USINT | Number of 16 bit words in Additional Status array. |
| Additional Status | Array of 16-bit words | Additional status. If Size of Additional Status is 0, there is no Additional Status. |
| Response Data | Array of octet | Response data per object definition from request. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Volume 1: Common Industrial Protocol (CIP™) Specification

# Chapter 3: Communication Object Classes


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Contents


3-1 Introduction ..... 5
3-1.1 Creating Connections Through the Connection Object ..... 5
3-1.2 Creating Connections Through the Connection Manager Object ..... 5
3-2 Link Producer Object Class Definition ..... 6
3-2.1 Link Producer Object Class Attributes ..... 6
3-2.2 Link Producer Object Class Services ..... 6
3-2.3 Link Producer Object Instance Attributes ..... 6
3-2.3.1 state, USINT data type ..... 6
3-2.3.2 connection_id ..... 6
3-2.4 Link Producer Object Instance Services ..... 6
3-2.5 Link Producer Instance Behavior ..... 7
3-3 Link Consumer Object Class Definition ..... 7
3-3.1 Link Consumer Object Class Attributes ..... 7
3-3.2 Link Consumer Class Services ..... 7
3-3.3 Link Consumer Instance Attributes ..... 7
3-3.3.1 state, USINT data type ..... 8
3-3.3.2 connection_id ..... 8
3-3.4 Link Consumer Instance Services ..... 8
3-3.5 Link Consumer Instance Behavior ..... 8
3-4 Connection Object Class Definition ..... 9
3-4.1 Connection Object Class Attributes ..... 9
3-4.2 Connection Object Class Services ..... 10
3-4.3 Connection Class Object Specific Services ..... 10
3-4.3.1 Connection Bind Service ..... 10
3-4.3.2 Producing Application Lookup Service ..... 11
3-4.4 Connection Object Instance Attributes ..... 12
3-4.4.1 State, Attribute 1 - USINT data type ..... 13
3-4.4.2 Instance_type, Attribute 2 - USINT data type ..... 14
3-4.4.3 TransportClass_trigger, Attribute 3 - BYTE data type ..... 14
3-4.4.3.1 Server Transport Class 0 and 1 Behavior ..... 16
3-4.4.3.2 Server Transport Class 2 Behavior ..... 17
3-4.4.3.3 Server Transport Class 3 Behavior ..... 18
3-4.4.3.4 Client Transport Class 0 and 1 Behavior: Cyclic ..... 19
3-4.4.3.5 Client Transport Class 2 and 3 Behavior: Cyclic ..... 20
3-4.4.3.6 Client Transport Class 0 and 1 Behavior: Change-Of-State ..... 20
3-4.4.3.7 Client Transport Class 2 and 3 Behavior: Change-Of-State ..... 21
3-4.4.3.8 Client Transport Class 0 and 1 Behavior: Application Object Triggered ..... 22
3-4.4.3.9 Client Transport Class 2 and 3 Behavior: Application Object Triggered ..... 23
3-4.4.3.10 TransportClass Trigger Summary ..... 24
3-4.4.4 DeviceNet_produced_connection_id, Attribute 4 – UINT data type ..... 25
3-4.4.5 DeviceNet_consumed_connection_id, Attribute 5 – UINT data type ..... 25
3-4.4.6 DeviceNet_initial_comm_characteristics, Attribute 6 – USINT data type ..... 25
3-4.4.7 Produced_connection_size, Attribute 7 – UINT data type ..... 25
3-4.4.8 Consumed_connection_size, Attribute 8 – UINT data type ..... 26
3-4.4.9 Expected_packet_rate, Attribute 9 – UINT data type ..... 26
3-4.4.10 CIP_produced_connection_id, Attribute 10 – UDINT data type ..... 27
3-4.4.11 CIP_consumed_connection_id, Attribute 11 – UDINT data type ..... 27
3-4.4.12 Watchdog_timeout_action, Attribute 12 – USINT data type ..... 27
3-4.4.13 Produced_connection_path_length, Attribute 13 – UINT data type ..... 27
3-4.4.14 Produced_connection_path, Attribute 14 – Packed EPATH data type ..... 28
3-4.4.15 Consumed_connection_path_length, Attribute 15 – UINT data type ..... 28
3-4.4.16 Consumed_connection_path, Attribute 16 – Packed EPATH data type ..... 28
3-4.4.17 Production_inhibit_time, Attribute 17 – UINT data type ..... 28



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

3-4.4.18 Connection_timeout_multiplier, Attribute 18 –USINT data type ..... 29
3-4.4.19 Connection_binding_list, Attribute 19 – Struct(UINT(size), Array of UINT[size]) data type ..... 29
3-4.5 Connection Timing ..... 29
3-4.5.1 Transmission Trigger Timer ..... 29
3-4.5.2 Inactivity/Watchdog Timer ..... 30
3-4.5.3 Production Inhibit Timer ..... 32
3-4.6 Connection Object Instance Services ..... 33
3-4.7 Connection Instance Behavior ..... 35
3-4.7.1 I/O Connection Instance Behavior ..... 35
3-4.7.2 CIP Bridged Connection Instance Behavior ..... 41
3-4.7.3 Explicit Messaging Connection Instance Behavior ..... 43
3-4.8 Connection Object Attribute Access Rules ..... 46
3-5 Connection Manager Object Class Definition ..... 49
3-5.1 Class Attributes ..... 49
3-5.2 Instance Attributes ..... 49
3-5.2.1 Semantics ..... 51
3-5.2.1.1 Percent CPU Utilization (Attribute 11) ..... 51
3-5.2.1.2 Max Connection Establishment Time (Attribute 14) ..... 51
3-5.2.1.3 I/O Packets per Second (Attribute 15) ..... 51
3-5.2.1.4 Percent I/O Utilization (Attribute 16) ..... 52
3-5.2.1.5 Explicit Packets Per Second (Attribute 17) ..... 52
3-5.2.1.6 Missed I/O Packets (Attribute 18) ..... 52
3-5.2.1.7 CIP I/O Connections (Attribute 19) ..... 52
3-5.2.1.8 CIP Explicit Connections (Attribute 20) ..... 52
3-5.3 Common Services ..... 52
3-5.3.1 Get_Attributes_All Response ..... 53
3-5.3.2 Set_Attributes_All Request ..... 54
3-5.4 Object-Specific Services ..... 54
3-5.4.1 Connection Manager Object Specific Service Parameters ..... 54
3-5.4.1.1 Network Connection Parameters ..... 55
3-5.4.1.2 Requested and Actual Packet Intervals ..... 57
3-5.4.1.3 Connection Serial Number ..... 60
3-5.4.1.4 Connection Timeout Multiplier ..... 60
3-5.4.1.5 Originator Vendor ID ..... 61
3-5.4.1.6 Originator Serial Number ..... 61
3-5.4.1.7 Connection Triad ..... 61
3-5.4.1.8 Connection Number ..... 61
3-5.4.1.9 Connection Path Size ..... 61
3-5.4.1.10 Connection Path ..... 61
3-5.4.1.11 Network Connection ID ..... 66
3-5.4.1.12 Transport Class and Trigger ..... 66
3-5.4.2 Forward_Open and Large_Forward_Open Request ..... 67
3-5.4.2.1 Non-Null Forward Open ..... 67
3-5.4.2.2 Null Forward Open ..... 69
3-5.4.3 Forward_Open and Large_Forward_Open Responses ..... 71
3-5.4.4 Forward_Close ..... 72
3-5.4.5 Unconnected_Send ..... 74
3-5.4.5.1 Unconnected_Send Request ..... 74
3-5.4.5.2 Unconnected_Send Response ..... 76
3-5.4.6 Get_Connection_Data ..... 77
3-5.4.7 Get_Connection_Owner ..... 78
3-5.4.8 Search_Connection_Data ..... 79
3-5.5 Connection Manager Object Instance Error Codes ..... 79
3-6 Application Connection Type using Class 0 or 1 Transports ..... 90
3-6.1 Real time formats including RUN/IDLE notification ..... 91
3-6.1.1 Modeless Format ..... 91


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

3-6.1.2 Zero Length Data Format ..... 91
3-6.1.3 Heartbeat Format ..... 92
3-6.1.4 32-Bit Header Format ..... 92
3-6.2 Configuration ..... 92
3-6.3 T→O Network Connection Rules ..... 93
3-6.3.1 T→O Multicast Matching Rules ..... 94
3-6.4 Specifying Different Application Connection Types ..... 94
3-6.5 Application types ..... 94
3-6.5.1 Listen Only ..... 95
3-6.5.2 Input Only ..... 95
3-6.5.3 Exclusive Owner ..... 95
3-6.5.4 Redundant Owner ..... 96
3-6.5.4.1 General ..... 96
3-6.5.4.2 Establishing the Connection ..... 96
3-6.5.4.3 Redundant Owner O→T Data Format ..... 97
3-6.5.4.4 Determining the owning connection ..... 97
3-6.5.4.5 Transporting events and data to a target application ..... 98
3-7 Port Object Class Definition ..... 99
3-7.1 Port Object Class Attributes ..... 100
3-7.2 Port Object Instance Attributes ..... 100
3-7.2.1 Semantics ..... 102
3-7.2.1.1 Port Type, Link Object and Port Type Name, Attributes 1, 3 and 5 ..... 102
3-7.2.1.2 Port Number, Attribute 2 ..... 103
3-7.2.1.3 Port Name, Attribute 4 ..... 103
3-7.2.1.4 Port Description, Attribute 6 ..... 104
3-7.2.1.5 Port Number and Node Address, Attribute 7 ..... 104
3-7.2.1.6 Port Node Range, Attribute 8 ..... 104
3-7.2.1.7 Chassis Identity, Attribute 9 ..... 104
3-7.2.1.8 Port Routing Capabilities, Attribute 10 ..... 105
3-7.2.1.9 Associated Communication Objects, Attribute 11 ..... 105
3-7.3 Common Services ..... 106
3-7.3.1 Get_Attributes_All Response ..... 106


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 3-1 Introduction

The CIP Communication Objects manage and provide the run-time exchange of messages. The Services, Attributes, and Behaviors associated with the Communication Objects are detailed in this Chapter. Part of an object definition involves assigning a Data Type to an attribute. See Appendix C for a detailed description of CIP Data Types and Data Management.

Important: It is not the intent of the following sections to specify any particular internal implementation.

The Communication Object Classes are defined by describing:

- Object Class Attributes
- Object Class Services
- Object Instance Attributes
- Object Instance Services
- Object Instance Behavior

Each CIP connection is represented by a Connection Object (Class code 0x05). The creation of this communication object resource can be done in one of two ways. Each subnet type defines which method shall be used. The two methods are:

- Use of the Create service (Service code 0x08) for the Connection Object
- Use of the Forward Open service for the Connection Manager Object

Subnet types may define other methods of creating this communication object resource. An example of this is DeviceNet's Allocate_Master/Slave_Connection_Set service. See the appropriate specification volume for details of subnet type-specific methods.

## 3-1.1 Creating Connections Through the Connection Object

When the subnet defines that connections are created through the Connection Object, a CIP device shall support the Create service for this class. The Create service instantiates a Connection Instance with attribute values defaulted as defined by the class. The connection instance is configured through individual access to each Connection instance attribute. A separate service request (Apply_Attributes, Service code 0x0D) is needed to transition the connection to the Established state.

## 3-1.2 Creating Connections Through the Connection Manager Object

When the subnet defines that connections are created through the Connection Manager Object, a CIP device shall support the Forward Open service of this class. When successful, the Connection Manager instantiates an instance of the Connection class. This connection instance is configured with the values sent in the Forward Open service and is transitioned to the established state. This single CIP service request is modeled internally as a single Connection Class service request (using the Create service) and several internal service requests (using the Set_Attribute_Single and Apply_Attributes services). A device supporting connection creation through the Connection Manager may or may not provide external visibility to the Connection Class instances.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 3-2 Link Producer Object Class Definition

The Link Producer Object is the component responsible for the low-level transmission of data.

**Important:** NO externally visible interface to the Link Producer Class across Explicit Messaging Connections exists. All services/attributes noted in the following sections describe internal behavior. These services/attributes are accessed via attributes and services of the Connection Object.

### 3-2.1 Link Producer Object Class Attributes

There are no Link Producer Class Attributes.

### 3-2.2 Link Producer Object Class Services

The services supported by the Link Producer Class are listed below.

- Create - Used internally to instantiate a Link Producer Object
- Delete - Used internally to deletes a Link Producer Object

### 3-2.3 Link Producer Object Instance Attributes

The following list describes the Link Producer Instance attributes.

#### 3-2.3.1 state, USINT data type

This contains the current state of the Link Producer instance. Possible states include the following:

**Table 3-2.1 Link Producer States**

| State Name | Description |
|---|---|
| Non-existent | The Link Producer has yet to be instantiated |
| Running | The Link Producer has been instantiated and is waiting to be told to transmit via the invocation of its Send service. |

#### 3-2.3.2 connection_id

The value placed within the message frame when this Link Producer is triggered to send. The placement and format of this value within the message frame is subnet type-specific. The Connection Object using this Link Producer internally initializes this attribute with the value in its **produced_connection_id** attribute. See section 3-4.4, Connection Instance Attributes for a definition of the produced_connection_id attributes.

### 3-2.4 Link Producer Object Instance Services

The services supported by a Link Producer Object Instance are listed below.

- Send - Used internally to tell the Link Producer to transmit data onto the subnet.
- Get_Attribute - Used internally to read a Link Producer Object attribute
- Set_Attribute - Used internally to modify a Link Producer Object attribute


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-2.5 Link Producer Instance Behavior

Figure 3-2.1 and Table 3-2.2 illustrate the Link Producer's Instance behavior.

**Figure 3-2.1 Link Producer in Object State Transition**

**Table 3-2.2 State/Event Matrix: Link Producer**

| Event | State | |
|---|---|---|
| | Non-Existent | Running |
| Class Create invoked internally | Class instantiates Link Producer Object. Link Producer enters the Running state. | Not applicable |
| Class Delete invoked internally | Error: Instance Not Present | Release all associated instance resources. Transition to the Non-existent state. |
| Send invoked internally | Error: Instance Not Present | Transmit the data |
| Set_Attribute invoked internally | Error: Instance Not Present | Modify the attribute |
| Get_Attribute invoked internally | Error: Instance Not Present | Return the attribute value |

## 3-3 Link Consumer Object Class Definition

The Link Consumer Object is the component responsible for the low-level reception of messages.

**Important:** NO externally visible interface to the Link Consumer Class across Explicit Messaging Connections exists. All services/attributes noted in the following sections describe internal behavior. These services/attributes are accessed via attributes and services of the Connection Object.

### 3-3.1 Link Consumer Object Class Attributes

There are no Link Consumer Class Attributes.

### 3-3.2 Link Consumer Class Services

The services supported by the Link Consumer Class are listed below.

- Create - Used internally to instantiate a Link Consumer Object
- Delete - Used internally to deletes a Link Consumer Object

### 3-3.3 Link Consumer Instance Attributes

The following list describes the Link Consumer Instance attributes.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-3.3.1 state, USINT data type

This contains the current state of the consumer instance. Possible states include:

**Table 3-3.1 Link Consumer States**

| State Name | Description |
|---|---|
| Non-existent | The Link Consumer has yet to be instantiated. |
| Running | The Link Consumer has been instantiated and is waiting to receive a message. |

### 3-3.3.2 connection_id

This attribute holds the Connection ID value in the message frame that denotes the message to be received by this consumer. The placement and format of this value within the message frame is subnet type-specific. The Connection Object utilizing this Link Consumer internally initializes this attribute with the value in its **consumed_connection_id** attribute. See section 3-4.4, Connection Instance Attributes for a definition of the consumed_connection_id attributes.

## 3-3.4 Link Consumer Instance Services

The services supported by a Link Consumer Object Instance are listed below.

- Get_Attribute - Used internally to read a Link Consumer Object attribute
- Set_Attribute - Used internally to modify a Link Consumer Object attribute

## 3-3.5 Link Consumer Instance Behavior

Figure 3-3.1 and Table 3-3.2 illustrate the Link Consumer's Instance behavior.

**Figure 3-3.1 Link Consumer Object State Transition Diagram**

**Table 3-3.2 State/Event Matrix: Link Consumer**

| Event | State | |
|---|---|---|
| | Non-Existent | Running |
| Class Create invoked internally | Class instantiates Link Consumer Object. Link Consumer enters the Running state. | Not applicable |
| Class Delete invoked internally | Error: Instance Not Present | Release all associated instance resources. Transition to the Non-existent state. |
| Data received | Not applicable | Deliver data to the associated Connection Object by invoking its Receive_Data() service |
| Set_Attribute invoked internally | Error: Instance Not Present | Modify the attribute |
| Get_Attribute invoked internally | Error: Instance Not Present | Return the attribute value |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 3-4 Connection Object Class Definition

**Class Code: 05 hex**

The Connection Class allocates and manages the internal resources associated with both I/O and Explicit Messaging Connections. The specific instance generated by the Connection Class is referred to as a *Connection Instance* or a *Connection* Object.

Unless otherwise noted, all services/attributes noted in the following sections are accessible using Explicit Messaging.

A Connection Object within a particular module actually represents one of the end-points of a Connection. It is possible for one of the Connection end-points to be configured and "active" (e.g. transmitting) without the other end-point(s) being present. Connection Objects are used to model the communication specific characteristics of a particular Application-to-Application(s) relationship.

A specific Connection Object Instance manages the communication-specific aspects related to an end-point. A CIP Connection Object uses the services provided by a Link Producer and/or Link Consumer to perform low-level data transmission and reception functions.

**Figure 3-4.1 Connection Object & Link Producer/Consumer Relationship**

## 3-4.1 Connection Object Class Attributes

The Connection Class attributes are defined below in Table 3-4.1.

**Table 3-4.1 Connection Class Attributes**

| Attribute ID | Need In Implementation | Access Rule | Attribute Name | Data Type | Attribute Description | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |
| 8 | Conditional [1] | | Connection Request Error Count | | See CIP Safety Specification (Volume 5, Chapter 5) | |
| 9 | Conditional [1] | | Safety Connection Counters | | | |

1 These attributes are not allowed if the device is not a safety device. If the device is a safety device, see Volume 5.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 3-4.2 Connection Object Class Services

The Connection Class supports the following CIP Common Services:

**Table 3-4.2 Connection Class Services**

| Service Code | Need In Implementation | Service Name | Service Description |
|---|---|---|---|
| 08hex | Optional | Create | Used to instantiate a Connection Object |
| 09hex | Optional | Delete | Used to delete all Connection Objects (independent of state) and to release all associated resources. When the Delete service is sent to the Connection Class (Instance ID set to zero (0)) versus a specific Connection Object Instance, then ALL Instances are deleted. |
| 05hex | Optional | Reset | Used to reset all resettable Connection Objects. The conditions under which a Connection is resettable are listed in the State Event Matrix in section 3-4.7 |
| 11hex | Optional | Find_Next_ Object_Instance | Used to search for Instance IDs associated with existing Connection Objects. The Connection Class returns the Instance ID associated with any Connection Object not in the **Non-Existent** state. |
| 0Ehex | Conditional | Get_Attribute_ Single | Used to read a Connection Class attribute value. This service is Required if any of the Connection Class Attributes are supported. |

## 3-4.3 Connection Class Object Specific Services

The following class specific services are defined for the connection object class.

**Table 3-4.3 Connection Class Object Specific Services**

| Service Code (hex) | Need In Implementation | Service Name | Service Description |
|---|---|---|---|
| 4Bhex | Conditional [1] | Connection Bind | Binds two connection instances |
| 4Chex | Conditional [1] | Producing Application Lookup | Find established connections that are producing data from the application object specified by this service |
| 4Ehex | Conditional [2] | SafetyClose | See CIP Safety Specification |
| 54hex | Conditional [2] | SafetyOpen | (Volume 5, Chapter 3) |

1 This service is required if the Create service is supported.
2 These services are not allowed if the device is not a safety device. If the device is a safety device, see Volume 5

### 3-4.3.1 Connection Bind Service

The Connection Bind object specific service binds two dynamically created I/O connection instances together for purposes of connection timeouts and deletions. A dynamically created I/O connection is the result of a Create service to the Connection class with the Instance_type attribute set to I/O (attribute value of 1). If one of these I/O connection instance times out or is deleted, the other instance shall exhibit the same behavior.

The Connection Bind service has the following request parameter:

**Table 3-4.4 Connection Bind Service Request Parameter**

| Parameter Name | Data Type | Parameter Description |
|---|---|---|
| Bound Instances | STRUCT of<br>UINT<br>UINT | Instance numbers of connection objects to be bound. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

There is no response data. The service request may return a status from the following list of status codes:

**Table 3-4.5 Connection Bind Service Status Codes**

| General Status Code | Extended Status Code | Status Description |
|---|---|---|
| 0x02 | 0x01 | One or both of the connection instances is non-existent. |
| 0x02 | 0x02 | The connection class and/or instance is out of resources to bind instances. |
| 0x0C | 0x01 | Both of the connection instances are existent, but at least one is not in the Established state. |
| 0x20 | 0x01 | Both connection instances are the same value. |
| 0xD0 | 0x01 | One or both of the connection instances is not a dynamically created I/O connection. |
| 0xD0 | 0x02 | One or both of the connection instances were created internally and the device is not allowing a binding to it. |

### 3-4.3.2 Producing Application Lookup Service

The Producing Application Lookup object specific service provides a mechanism to find one or more connection instances in the Established state producing data from a given application object. If the requested producing application path is being produced by any connection in the Established state, the instance number of each connection producing from that path is returned.

The Producing Application Lookup service has the following service request parameter:

**Table 3-4.6 Producing Application Lookup Service Request Parameter**

| Parameter Name | Data Type | Parameter Description |
|---|---|---|
| Producing Application Path | Packed EPATH | Connection path of producing application to be searched for within connections in the Established state. The path shall be a single Logical or Symbolic path. |

The Producing Application Lookup service has the following service response parameters:

**Table 3-4.7 Producing Application Lookup Service Response Parameter**

| Parameter Name | Data Type | Parameter Description |
|---|---|---|
| Instance Count | UINT | Number of Instances returned in the Connection Instance List parameter. |
| Connection Instance List | Struct of UINT | List of instance numbers of connection producing the data from the requested connection path within the node. |

The service request may return a status from the following list of status codes:

**Table 3-4.8 Producing Application Lookup Service Status Codes**

| General Status Code | Extended Status Code | Status Description |
|---|---|---|
| 0x02 | 0x01 | The connection path was not found in any connection instance in the Established state. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 3-4.4 Connection Object Instance Attributes

The following list provides a summary of the Connection Instance attributes and their associated data types.

**Table 3-4.9 Connection Object Instance Attributes**

| Attr ID | Need In Implementation | Attribute Name | Data Type | Brief Description of Attribute |
|---|---|---|---|---|
| 1 | Required | State | USINT | State of the object |
| 2 | Required | Instance_type | USINT | Indicates either I/O or Messaging Connection |
| 3 | Required | TransportClass_trigger | BYTE | Defines behavior of the Connection |
| 4 | Conditional | DeviceNet_produced_ connection_id | UINT | Placed in CAN Identifier Field when the Connection transmits on a DeviceNet subnet. Described in Vol 3, DeviceNet Adapation of CIP. |
| 5 | Conditional | DeviceNet_consumed_ connection_id | UINT | CAN Identifier Field value that denotes message to be received on a DeviceNet subnet. Described in Vol 3, DeviceNet Adaptation of CIP. |
| 6 | Conditional | DeviceNet_initial_comm_ characteristics | BYTE | Defines the Message Group(s) across which productions and consumptions associated with this Connection occur on a DeviceNet subnet. Described in Vol 3, DeviceNet Adaptation of CIP. |
| 7 | Required | Produced_connection_size | UINT | Maximum number of bytes transmitted across this Connection |
| 8 | Required | Consumed_connection_size | UINT | Maximum number of bytes received across this Connection |
| 9 | Required | Expected_packet_rate | UINT | Defines timing associated with this Connection |
| 10 | Conditional | CIP_produced_connection_id | UDINT | Identifies the message sent on the subnet by this connection. |
| 11 | Conditional | CIP_consumed_connection_id | UDINT | Identifies the message received from the subnet for this connection. |
| 12 | Required | Watchdog_timeout_action | USINT | Defines how to handle Inactivity/Watchdog timeouts |
| 13 | Required | Produced_connection_path_ length | UINT | Number of bytes in the produced_connection_path attribute |
| 14 | Required | Produced_connection_path | Packed EPATH | Specifies the Application Object(s) whose data is to be produced by this Connection Object. See Appendix C. |
| 15 | Required | Consumed_connection_path_ length | UINT | Number of bytes in the consumed_connection_path attribute |
| 16 | Required | Consumed_connection_path | Packed EPATH | Specifies the Application Object(s) that are to receive the data consumed by this Connection Object. See Appendix C. |
| 17 | Conditional | Production_inhibit_time | UINT | Defines minimum time between new data production. This attribute is required for all I/O Client connections, except those with a production trigger of Cyclic. |
| 18 | Optional | Connection_timeout_multiplier | USINT | Specifies the multiplier applied to the expected_packet_rate value to derive the value for the Inactivity/Watchdog Timer. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need In Implementation | Attribute Name | Data Type | Brief Description of Attribute |
|---|---|---|---|---|
| 19 | Conditional | Connection_binding_list | STRUCT of<br>UINT<br>Array of<br>UINT | List of I/O connection instances bound to this instance. |

**Important:** Access Rules for the Connection Object Instance Attributes are defined in section 3-4.8.

The list below provides detailed descriptions of the Connection Object Instance Attributes. The defined ***default*** attribute values are to be used when no other internal and/or system-defined rules exist.

### 3-4.4.1 State, Attribute 1 - USINT data type

This attribute defines the current state of the Connection instance. Table 3-4.10 defines the possible states and assigns a value used to indicate that state. Also see Figure 3-4.17 and Figure 3-4.20 for state transition behavior.

**Table 3-4.10 Values assigned to the state attribute**

| Value | State Name | Description |
|---|---|---|
| 00 | Non-existent | The Connection has yet to be instantiated |
| 01 | Configuring | The Connection has been instantiated and is waiting for the following events to occur: (1) to be properly configured and (2) to be told to apply the configuration. |
| 02 | Waiting For Connection ID[2] | The Connection instance is waiting exclusively for its consumed_connection_id and/or produced_connection_id attribute to be set.[1] |
| 03 | Established | The Connection has been validly/fully configured and the configuration has been successfully applied. |
| 04 | Timed Out | If a Connection Object experiences an Inactivity/Watchdog timeout, then a transition may be made to this state. See the watchdog_timeout_action attribute description and the description of the Inactivity/Watchdog Timer (section 3-4.5) for more details. |
| 05 | Deferred Delete[2] | If an Explicit Messaging Connection Object experiences an Inactivity/Watchdog timeout, then a transition may be made to this state. See the watchdog_timeout_action attribute description and the description of the Inactivity/Watchdog Timer (section 3-4.5) for more details. |
| 06 | Closing | A CIP Bridged connection object has received, and is processing, a Forward Close from the Connection Manager. The deletion of the connection does not occur until after a successful Forward Close response has been received from the target node. |

1 When the Connection instance attributes are applied it may not be possible for the module to generate the produced_connection_id and/or consumed_connection_id values (see initial_comm_characteristics attribute for rules). If this is the case then all the tasks required to apply the attributes are performed except for the initialization of these attributes within the Connection and associated Link Producer/Consumer Objects and a transition is made to this state. In this state the Connection instance is waiting exclusively for its produced_connection_id and/or consumed_connection_id attributes to be set.

2 This value is only used on DeviceNet.

**Important:** A dynamically created connection instance is the child of the Explicit Messaging connection across which it was created. Different subnet types may provide other mechanisms for creating a connection that imply a particular parent-child relationship. See network-specific specifications for details.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Important:** All resources associated with a Connection Instance (*A)* that has been dynamically created across an Explicit Messaging Connection (*B*) must be released if the Explicit Messaging Connection (*B)* times out prior to the dynamically created Connection Instance (*A)* transitioning to the Established state.

**Important:** When a transition is made to the Established state, all timers associated with the Connection Object are activated (see section 3-4.5, Connection Timing)

### 3-4.4.2 Instance_type, Attribute 2 - USINT data type

This attribute defines the instance type. See Table 3-4.11.

**Table 3-4.11 Values assigned to the instance_type attribute**

| Value | Meaning |
|---|---|
| 00 | Explicit Messaging. This Connection Instance represents one of the end-points of an Explicit Messaging Connection. An Explicit Messaging Connection is dynamically created by sending the *Open Explicit Messaging Connection Request* to the Connection Class. |
| 01 | I/O. This Connection Instance represents one of the end-points of an I/O Connection. An I/O Connection is dynamically created by sending a *Create Request* to the Connection Class. |
| 02 | CIP Bridged. This Connection Instance represents an intermediate 'hop' of a bridged I/O or Explicit Messaging connection. A pair of CIP Bridged connection objects (one for each subnet bridged between) are dynamically created by a node after successfully receiving a Forward Open service to the Connection Manager object when this node is not the end point (target). See Chapter 10 for additional information. |

### 3-4.4.3 TransportClass_trigger, Attribute 3 - BYTE data type

Defines whether this is a producing only, consuming only, or both producing and consuming connection. If this end point is to perform a data production, this attribute also defines the event that triggers the production. The eight (8) bits are divided as follows:

**Figure 3-4.2 Transport Class Trigger Attribute**

The ***Direction bit*** of the transportClass_trigger byte indicates whether the end-point is to act as the Client or the Server on this connection. The following values are defined:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-4.12 Possible Values within Direction Bit**

| Value | Meaning | |
|---|---|---|
| 0 | Client | This end-point provides the Client behavior associated with this Connection. Additionally, this value indicates that the *Production Trigger* bits within the transportClass_trigger byte contain the description of when the Client is to produce the message associated with this connection. Client connections with production trigger value of 0 or 1 (Cyclic or Change-of-State) shall produce immediately after transitioning to the Established state. |
| 1 | Server | This end-point provides the Server behavior associated with this Connection. In addition, this value indicates that the *Production Trigger* bits within the transportClass_trigger byte are to be IGNORED. The *Production Trigger* bits are ignored due to the fact that a Server end-point *reacts* to the transmission from the Client. The only means by which a Server end-point is triggered to transmit is when this *reaction* calls for the production of a message (Transport Classes 2 or 3). |

The following table lists the values that are possible within the ***Production Trigger*** bits of the **transportClass_trigger** attribute.

**Table 3-4.13 Possible Values within Production Trigger Bits**

| If the value is: | Then the Production of a message is: | |
|---|---|---|
| 0 | Cyclic | The expiration of the Transmission Trigger Timer triggers the data production. See section 3-4.5, Connection Timing for a detailed description of the Transmission Trigger Timer. |
| 1 | Change-Of-State | Production occurs when a change-of-state is detected by the Application Object. The term used to describe this production is “new data”. Note that the consuming end-point may have been configured to expect the packet at a certain rate, regardless of the triggering mechanism at the producing end-point. See the description of the expected_packet_rate attribute of a Connection Object and the description of Connection Timing in section 3-4.5 for more information. |
| 2 | Application Object Triggered | The Application Object decides when to trigger the production. The term used to describe this production is “new data”. Note that the consuming end-point may have been configured to expect the packet at a certain rate, regardless of the triggering mechanism at the producing end-point. See the description of the expected_packet_rate attribute of a Connection Object and the description of Connection Timing in section 3-4.5 for more information. |
| 3 - 7 | Reserved by CIP | |

Table 3-4.14 lists possible values within the ***Transport Class*** nibble of the **transportClass_trigger** attribute. Behaviors resulting from these particular values are illustrated in the series of figures that follow the table.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-4.14 Possible Values within Transport Class Bits**

| Value | Meaning | |
|---|---|---|
| 0 | Transport Class 0 | Based on the value within the *Dir* bit, this connection end-point will be a producing only OR consuming only end-point. Upon application of this Connection instance, the module instantiates either a Link Producer (*Dir* bit = Client, producing only) or a Link Consumer (*Dir* bit = Server, consuming only) to be associated with this Connection. |
| 1 | Transport Class 1 | |
| 2 | Transport Class 2 | Indicates that the module will both produce AND consume across this connection. The Client end-point generates the first data production that is consumed by the Server, which causes the Server to return a production that is consumed by the Client. |
| 3 | Transport Class 3 | |
| 4 | Transport Class 4 | Non-blocking |
| 5 | Transport Class 5 | Non-blocking, fragmenting |
| 6 | Transport Class 6 | Multicast, fragmenting |
| 7 - F | Reserved | |

A 16-bit sequence count value is prepended to all Class 1, 2, and 3 transports. This value is used to detect delivery of duplicate data packets. Sequence count values are initialized on the first message production and determined by the transport and trigger type on subsequent message productions. A resend of old data to maintain the connection shall not cause the sequence count to change, and a consumer shall ignore data when it is received with a duplicate sequence count. Consuming applications can use this mechanism to distinguish between new samples and old samples that were sent to maintain the connection.

For Class 2 and 3 transports, when a request is received with a duplicate sequence count the receiving device shall respond with the original response. This mechanism can be used to keep Class 2 and 3 transports alive and to deal with cases of dropped packets.

The following tables and figures illustrate the valid combinations of Production Trigger and Transport Class and provides a description of the Client and Server behaviors. See section 3-4.5.1 for a description of the Transmission Trigger Timer, which is shown in the illustrations.

### 3-4.4.3.1 Server Transport Class 0 and 1 Behavior

**Figure 3-4.3 Server Transport Class 0 and 1 Behavior**

| Direction Bit | Production Trigger Bits | Transport Class Bits | Meaning |
|---|---|---|---|
| 1 | X (ignored) | 0 | Server Transport Class 0. |
| 1 | X (ignored) | 1 | **Server Transport Class 1.** The message received is prepended with a 16-bit sequence count value. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-4.4.3.2 Server Transport Class 2 Behavior

**Figure 3-4.4 Server Transport Class 2 Connection Object**

| Direction Bit | Production Trigger Bits | Transport Class Bits | Meaning |
|---|---|---|---|
| 1 | X (ignored) | 2 | **Server Transport Class 2.** The act of consumption by the Link Consumer IMMEDIATELY triggers the associated Link Producer to transmit without intervention by the application. The message received is prepended with a 16-bit sequence count value. The message returned is prepended with this same 16-bit sequence count value. |

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-4.4.3.3 Server Transport Class 3 Behavior

**Figure 3-4.5 Server Transport Class 3 Behavior**

| Direction Bit | Production Trigger Bits | Transport Class Bits | Meaning |
|---|---|---|---|
| 1 | X (ignored) | 3 | **Server Transport Class 3.** When the Link Consumer receives a message, it delivers it to the Application Object specified within the **consumed_connection_path** attribute. The Application Object then validates this *receive data* event (illustrated as #2a in the figure below). If the Application Object determines that the *receive data* event is valid, then it is REQUIRED to trigger a production as illustrated below. If the Application detects an error, then it may or may not trigger a production based on its own internal logic. The message received is prepended with a 16-bit sequence count value. The message returned is prepended with this same 16-bit sequence count value. |

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-4.4.3.4 Client Transport Class 0 and 1 Behavior: Cyclic

**Figure 3-4.6 Client Transport Class 0 and 1 Behavior: Cyclic**

| Direction Bit | Production Trigger Bits | Transport Class Bits | Meaning |
|---|---|---|---|
| 0 | 0 | 0 | Client Transport Class 0: Cyclic.<br>Message production is triggered by the Transmission Trigger Timer |
| 0 | 0 | 1 | Client Transport Class 1: Cyclic.<br>The message sent is prepended with a 16-bit sequence count value.<br>The sequence count is incremented only when the application updates the application production data.<br>Message production is triggered by the Transmission Trigger Timer. |

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-4.4.3.5 Client Transport Class 2 and 3 Behavior: Cyclic

**Figure 3-4.7 Client Transport Classes 2 & 3 Behavior: Cyclic**

| Direction Bit | Production Trigger Bits | Transport Class Bits | Meaning |
|---|---|---|---|
| 0 | 0 | 2 | Client Transport Class 2: Cyclic |
| 0 | 0 | 3 | Client Transport Class 3: Cyclic |

A 16-bit sequence count value is prepended to the message before transmission.
The sequence count is incremented only when the application updates the application production data.
The message consumed due to this production is also prepended with a 16 bit sequence count with a value equal to that of the sequence count sent.
Message production is triggered by the Transmission Trigger Timer.

### 3-4.4.3.6 Client Transport Class 0 and 1 Behavior: Change-Of-State

**Figure 3-4.8 Client Transport Class 0 and 1 Behavior: Change-Of-State**

| Direction Bit | Production Trigger Bits | Transport Class Bits | Meaning |
|---|---|---|---|
| 0 | 1 | 0 | Client Transport Class 0: Change-of-State.<br>Message production is triggered when the application production data has changed and by the Transmission Trigger Timer. |
| 0 | 1 | 1 | Client Transport Class 1: Change-of-State.<br>The message sent is prepended with a 16-bit sequence count value.<br>The sequence count is incremented only when the application production data is changed.<br>Message production is triggered when the application production data has changed and by the Transmission Trigger Timer. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-4.4.3.7 Client Transport Class 2 and 3 Behavior: Change-Of-State

**Figure 3-4.9 Client Transport Classes 2 & 3 Behavior: Change-Of-State**

| Direction Bit | Production Trigger Bits | Transport Class Bits | Meaning |
|---|---|---|---|
| 0 | 1 | 2 | Client Transport Class 2: Change-of-State |
| 0 | 1 | 3 | Client Transport Class 3: Change-of-State |

A 16-bit sequence count value is prepended to the message before transmission.
The sequence count is incremented only when the application production data is changed.
The message consumed due to this production is also prepended with a 16-bit sequence count with a value equal to that of the sequence count sent.
Message production is triggered when the application production data has changed and by the Transmission Trigger Timer.

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-4.4.3.8 Client Transport Class 0 and 1 Behavior: Application Object Triggered

**Figure 3-4.10 Client Transport Class 0 and 1 Behavior: Application Object Triggered**

| Direction Bit | Production Trigger Bits | Transport Class Bits | Meaning |
|---|---|---|---|
| 0 | 2 | 0 | Client Transport Class 0: Application Object Triggered.<br>Message production is triggered when the application updates the application production data and by the Transmission Trigger Timer. |
| 0 | 2 | 1 | Client Transport Class 1: Application Object Triggered.<br>The message sent is prepended with a 16-bit sequence count value.<br>The sequence count is incremented only when the application updates the application production data.<br>Message production is triggered when the application updates the application production data and by the Transmission Trigger Timer. |

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-4.4.3.9 Client Transport Class 2 and 3 Behavior: Application Object Triggered

**Figure 3-4.11 Client Transport Classes 2 & 3 Behavior: Application Object Triggered**

| Direction Bit | Production Trigger Bits | Transport Class Bits | Meaning |
|---|---|---|---|
| 0 | 2 | 2 | Client Transport Class 2: Application Object Triggered |
| 0 | 2 | 3 | Client Transport Class 3: Application Object Triggered |

A 16-bit sequence count value is prepended to the message before transmission.
The sequence count is incremented only when the application updates the application production data.
The message consumed due to this production is also prepended with a 16-bit sequence count with a value equal to that of the sequence count sent.
Message production is triggered when the application updates the application production data and by the Transmission Trigger Timer.

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-4.4.3.10 TransportClass Trigger Summary

Table 3-4.15 summarizes the valid values for the **transportClass_trigger** attribute of a Connection Instance.

**Table 3-4.15 Transport Class_Trigger Attribute Value Summary**

| TransportClass_trigger bits | Meaning |
|---|---|
| 1 xxx 0000 | Direction = Server, Production Trigger = IGNORED, Transport Class = 0. |
| 1 xxx 0001 | Direction = Server, Production Trigger = IGNORED, Transport Class = 1. |
| 1 xxx 0010 | Direction = Server, Production Trigger = IGNORED, Transport Class = 2. |
| 1 xxx 0011 | Direction = Server, Production Trigger = IGNORED, Transport Class = 3. This is the value assigned to this attribute within the Server end-point of an Explicit Messaging Connection. |
| 0 000 0000 | Direction = Client, Production Trigger = Cyclic, Transport Class = 0. |
| 0 000 0001 | Direction = Client, Production Trigger = Cyclic, Transport Class = 1. |
| 0 000 0010 | Direction = Client, Production Trigger = Cyclic, Transport Class = 2. |
| 0 000 0011 | Direction = Client, Production Trigger = Cyclic, Transport Class = 3. |
| 0 001 0000 | Direction = Client, Production Trigger = Change-Of-State, Transport Class = 0. |
| 0 001 0001 | Direction = Client, Production Trigger = Change-Of-State, Transport Class = 1. |
| 0 001 0010 | Direction = Client, Production Trigger = Change-Of-State, Transport Class = 2. |
| 0 001 0011 | Direction = Client, Production Trigger = Change-Of-State, Transport Class = 3. |
| 0 010 0000 | Direction = Client, Production Trigger = Application Object, Transport Class = 0. |
| 0 010 00001 | Direction = Client, Production Trigger = Application Object, Transport Class = 1 |
| 0 010 0010 | Direction = Client, Production Trigger = Application Object, Transport Class = 2. |
| 0 010 0011 | Direction = Client, Production Trigger = Application Object, Transport Class = 3. This is the value assigned to this attribute within the Client end-point of an Explicit Messaging Connection. |
| 1 111 1111 | Default value assigned to this attribute within an I/O Connection. |

Table 3-4.16 summarizes Figure 3-4.6, Figure 3-4.8 and Figure 3-4.10 pertaining to client behavior for Class 0 in regards to message production for the different trigger types.

**Table 3-4.16 Transport Class 0 Client Behavior Summary**

| Trigger Type | Production Triggered By | Repeated Production to Maintain Connection |
|---|---|---|
| Cyclic | Transmission Timer (API) | n/a |
| Change of State | Detected data change | Transmission Timer (API) |
| Application | Application update of data | Transmission Timer (API) |

Table 3-4.17 summarizes Figure 3-4.6 through Figure 3-4.11 pertaining to client behavior for Classes 1, 2 and 3 in regards to message production and Sequence Count for the different trigger types.

**Table 3-4.17 Transport Class 1, 2 and 3 Client Behavior Summary**

| Trigger Type | Sequence Count Incremented By | Production Triggered By | Repeated Production to Maintain Connection |
|---|---|---|---|
| Cyclic | Application update of data | Transmission Timer (API) | n/a |
| Change of State | Detected data change | Detected data change | Transmission Timer (API) |
| Application | Application update of data | Application update of data | Transmission Timer (API) |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-4.4.4 DeviceNet_produced_connection_id, Attribute 4 – UINT data type

This attribute is Required for a DeviceNet subnet. Other subnet types shall not use this attribute. See Volume 3, DeviceNet Adaptation of CIP, Chapter 3 for an explanation of this attribute.

### 3-4.4.5 DeviceNet_consumed_connection_id, Attribute 5 – UINT data type

This attribute is Required for a DeviceNet subnet. Other subnet types shall not use this attribute. See Volume 3, DeviceNet Adaptation of CIP, Chapter 3 for an explanation of this attribute.

### 3-4.4.6 DeviceNet_initial_comm_characteristics, Attribute 6 – USINT data type

This attribute is Required for a DeviceNet subnet. Other subnet types shall not use this attribute. See Volume 3, DeviceNet Adaptation of CIP, Chapter 3 for an explanation of this attribute.

### 3-4.4.7 Produced_connection_size, Attribute 7 – UINT data type

The meaning of this attribute is different for Explicit Messaging Connections than it is for I/O Connections. If the subnet defines a fragmentation protocol and the device supports fragmentation, this size may be larger than the largest frame size. See the network specific adaptation specifications for more details.

For Explicit Messaging Connections:

This attribute signifies the maximum number of Message Router Request/Response Data octets (See CIP Common, Chapter 2) that a device is able to transmit across this Connection. Devices that place a known limit on the maximum amount of Message Router Request/Response Data that can be transmitted in a single message, or single fragmented series initialize this attribute accordingly. Devices that cannot or do not predefine an up-front transmit limit place the value 0xffff into this attribute (there may still be a limit, however, it is not known in advance).

**Important:** Due to the nature of Explicit Messaging, the length of Explicit Messages will fluctuate over the lifetime of a connection. Explicit Messaging Connections perform fragmentation based on the *length* of the current message to transmit where the subnet type may define a fragmentation protocol.

***For I/O Connections:***

If the transportClass_trigger indicates that this Connection instance is to produce, then this attribute defines the maximum amount of I/O data that may be produced as a single unit across this connection. The amount of I/O to be transmitted at any given point in time can be less than or equal to the connection_size attribute.

This attribute defaults to zero (0) within an I/O Connection. If the subnet type supports fragmentation and this attribute is set to a value greater than the largest payload in an I/O Connection, then the Connection will break up the data into multiple fragments. See the network specific adaptation specifications for more details.

**Important:** Fragmentation within I/O Connections is performed based on the value within this attribute, regardless of the current amount of data to transmit.

**Important:** I/O Messages that contain no Application I/O Data and were configured to contain data (via produced_connection_size being greater than zero (0)) are defined to indicate a *No Data* event for the receiving Application Object(s). The behavior of an Application Object upon detection of the *No Data* event is Application Object specific.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-4.4.8 Consumed_connection_size, Attribute 8 – UINT data type

The meaning of this attribute is different for Explicit Messaging Connections than it is for I/O Connections. If the subnet defines a fragmentation protocol and the device supports fragmentation, this size may be larger than the largest frame size. See the network specific adaptation specifications for more details.

For Explicit Messaging Connections:

This attribute signifies the maximum number of Message Router Request/Response Data octets (See Chapter 2) that a device is able to receive across this Connection.

Devices that place a known limit on the maximum amount of Message Router Request/Response Data that can be received in a single message, or single fragmented series initialize this attribute accordingly. Devices that cannot or do not predefine an up-front receive limit place the value 0xffff into this attribute (there may still be a limit, however, it is not known in advance). Because of the nature of Explicit Messaging, the length of Explicit Messages will fluctuate over the lifetime of a connection.

***For I/O Connections:***

If the transportClass_trigger attribute indicates that this Connection is to consume, then this attribute defines the maximum amount of data that may be received as a single unit across this connection. The actual amount of I/O data received at any given time can be less than or equal to the connection_size attribute.

This attribute defaults to zero (0) within an I/O Connection. If the subnet type supports fragmentation and this attribute is set to a value greater than the largest payload in an I/O Connection, then the Connection will process the fragmentation protocol. See the network specific adaptation specifications for more details.

The length of an I/O Message must be less than or equal to this attribute for an I/O Connection Object to receive it as a valid message. If an I/O Connection Object receives a message whose length is greater than this attribute, then it immediately discards the message and discontinues any subsequent processing. Note that with respect to the Server end-point of a Transport Class 2 or 3 Connection, this too much data error condition results in no response being transmitted and the watchdog timer is not kicked.

### 3-4.4.9 Expected_packet_rate, Attribute 9 – UINT data type

This attribute is used to generate the values loaded into the *Transmission Trigger Timer* and the *Inactivity/Watchdog Timer*. See section 3-4.5, Connection Timing for a description of the Transmission Trigger and Inactivity/Watchdog timers.

The resolution of this attribute is in milliseconds. A request to configure this attribute may result in the specification of a time value that a product cannot meet. In addition to performing product specific range checking when a request to modify this attribute is received, the following steps are performed:

- If the specified value is not equal to an increment of the available clock resolution, then the value is rounded **up** to the next serviceable value. For example: a Set_Attribute_Single request is received specifying the value 5 for the **expected_packet_rate** attribute and the product provides a 10 millisecond resolution on timers. In this case the product would load the value 10 into the **expected_packet_rate** attribute.
- The value that is actually loaded into the **expected_packet_rate** attribute is reported in the Service Data Field of a Set_Attribute_Single response message associated with a request to modify this attribute.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- If the requested value is equal to an increment of the clock resolution, then the requested value is loaded into the expected_packet_rate and reported in the response. For example: if the value 100 is requested and the clock resolution is 10 milliseconds, then of a value of 100 is loaded.
- When a Connection Object is in the Established state, any modifications to the **expected_packet_rate** attribute have immediate effect on the Inactivity/Watchdog Timer. The following steps are performed by a Connection Object in the Established state when a request is received to modify the **expected_packet_rate** attribute:
  - the current Inactivity/Watchdog Timer is canceled
  - a new Inactivity/Watchdog Timer is activated based on the new value in the expected_packet_rate attribute.

This attribute defaults to 2500 (2500 milliseconds) within Explicit Messaging Connections, and to zero (0) within an I/O Connection.

### 3-4.4.10 CIP_produced_connection_id, Attribute 10 – UDINT data type

Contains the Connection ID, which identifies messages to be sent across this connection (if any). This attribute shall not be implemented when the subnet type is DeviceNet.

### 3-4.4.11 CIP_consumed_connection_id, Attribute 11 – UDINT data type

Contains the Connection ID, which identifies messages to be received across this connection (if any). This attribute shall not be implemented when the subnet type is DeviceNet.

### 3-4.4.12 Watchdog_timeout_action, Attribute 12 – USINT data type

This attribute defines the action the Connection Object should perform when the Inactivity/Watchdog Timer expires. The table below defines the specifics of this attribute.

**Table 3-4.18 Values for the watchdog_timeout_action**

| Value | Meaning |
|---|---|
| 0 | *Transition to Timed Out*. The Connection transitions to the Timed Out state and remains in this state until it is Reset or Deleted. The command to Reset or Delete could come from an internal source (e.g., an Application Object) or could come from the network (e.g., a configuration tool). This is the default value for this attribute with respect to I/O Connections. This value is invalid for Explicit Messaging Connections. |
| 1 | *Auto Delete*. The Connection Class automatically deletes the Connection if it experiences an Inactivity/Watchdog timeout. This is the default value for this attribute with respect to Explicit Messaging Connections. |
| 2 | *Auto Reset*. The Connection remains in the Established state and immediately restarts the Inactivity/Watchdog timer. This value is invalid for Explicit Messaging Connections. |
| 3 | *Deferred Delete*. The Connection transitions to the Deferred state if any child connection instances are in the Established state. If no child connection instances are in the Established state the connection is deleted. This value is only used on DeviceNet and is invalid for I/O Messaging Connections. |
| 4 - FF | Reserved by CIP |

### 3-4.4.13 Produced_connection_path_length, Attribute 13 – UINT data type

Specifies the number of bytes of information within the **produced_connection_path** attribute. This is automatically initialized when the **produced_connection_path** attribute is configured. This attribute defaults to the value zero (0).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-4.4.14 Produced_connection_path, Attribute 14 – Packed EPATH data type

The **produced_connection_path** attribute is made up of a byte stream which defines the Application Object(s) whose *data* is to be produced by this Connection Object. **The format of this byte stream is specified in Appendix C, Abstract Syntax Encoding for Segment Types.** This attribute defaults to being empty upon instantiation of the Connection. It remains empty within Explicit Messaging Connections and within Connection Objects that do not produce.

### 3-4.4.15 Consumed_connection_path_length, Attribute 15 – UINT data type

Specifies the number of bytes of information within the **consumed_connection_path** attribute. This is automatically initialized when the **consumed_connection_path** attribute is configured. This attribute defaults to the value zero (0).

### 3-4.4.16 Consumed_connection_path, Attribute 16 – Packed EPATH data type

The **consumed_connection_path** attribute is made up of a byte stream which defines the Application Object(s) that are to receive the *data* consumed by this Connection Object. **The format of this byte stream is specified in Appendix C, Abstract Syntax Encoding for Segment Types.** This attribute defaults to being empty upon instantiation of the Connection. It remains empty within Explicit Messaging Connections and within Connection Objects that do not consume.

### 3-4.4.17 Production_inhibit_time, Attribute 17 – UINT data type

This attribute is used to configure the minimum delay time between new data production. This is required for all I/O Client connections, except those with a production trigger of Cyclic. The Set_Attribute_Single service must be supported when this attribute is implemented. A value of zero (the default value for this attribute) indicates no inhibit time.

The resolution of this attribute is in milliseconds. A request to configure this attribute may result in the specification of time value that a product cannot meet. In addition to performing product specific range checking when a request to modify this attribute is received, the following steps are performed:

- If the specified value is not equal to an increment of the available clock resolution, then the value is rounded up to the next serviceable value. For example: a Set_Attribute_Single request is received specifying the value 5 for the production_inhibit_time attribute and the product provides a 10 millisecond resolution on times. In this case the product would load the value 10 into the production_inhibit_time attribute.
- The value that is actually loaded into the production_inhibit_time attribute is reported in the Service Data Field of a Set_Attribute_Single response message associated with a request to modify this attribute.
- If the requested value is equal to an increment of the clock resolution, then the requested value is loaded into the production_inhibit_time and reported in the response. For example: the value 100 is requested and the clock resolution is 10 milliseconds.

The Production Inhibit Timer shall be restarted each time new data production occurs. Data produced due to the expiration of the transmission trigger timer shall not result in the Production Inhibit Timer being restarted.

When a Connection Object is in the **Established** state, any modifications to the **production_inhibit_time** attribute have no effect on a currently running Production Inhibit Timer. The new production_inhibit_time value is loaded into the Production Inhibit Timer on the following new data production.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

When the apply_attributes service is received, the **production_inhibit_time** must beverified against the **expected_packet_rate** attribute. If the **expected_packet_rate v**alue is greater than zero, but less than the **production_inhibit_time** value, then an error shall be returned. In this case, where two attribute values conflict use the **production_inhibit_time attribute** ID as the additional error code returned in the error response.

#### 3-4.4.18 Connection_timeout_multiplier, Attribute 18 –USINT data type

The Connection_timeout_multiplier specifies the multiplier applied to the expected packet rate to obtain the value used by the Inactivity/Watchdog Timer. See the Connection Timeout Multiplier parameter description within the Connection Manager Object Specific Service Parameters for the enumerated values of this attribute. The default value for this attribute is zero (specifying a multiplier of 4).

#### 3-4.4.19 Connection_binding_list, Attribute 19 – Struct(UINT(size), Array of UINT[size]) data type

The Connection_binding_list attribute identifies connection instances that are bound to this connection. The attribute structure provides a 16 bit count of bound connections, followed by a list of the bound instances. This attribute shall be supported if the Connection Bind service is supported.

### 3-4.5 Connection Timing

Three types of timers are involved in a connection:

- Transmission Trigger Timer
- Inactivity/Watchdog Timer
- Production Inhibit Timer

The first two timers are initialized based on the value in the **expected_packet_rate** attribute.

**Important:** For Explicit Messaging, the Application is responsible for providing *response timeout* facilities. The amount of time a Client waits for a Server to respond to a request depends on the application and, possibly, on the service. Due to Media Access mechanisms used for accessing the subnet, an implementation should wait until an Explicit Request Message is transmitted before activating any response related timers.

#### 3-4.5.1 Transmission Trigger Timer

The Transmission Trigger Timer is required to be implemented by the client endpoint of an I/O Connection. It is optional for the client endpoint of an explicit messaging connection. It shall not be implemented in the server endpoint of any connection.

The Transmission Trigger Timer value shall be equal to the **expected packet rate**. Expiration of this timer shall result in the production of the associated data regardless of trigger type or connection class.

The Transmission Trigger Timer/expected_packet_rate establishes the production rate of cyclic connections, and the heartbeat/keep-alive production rate for Change-of-State and application triggered connections.

The Transmission Trigger Timer is restarted with the **expected_packet_rate** value immediately upon the Connection Object being told to produce new data by the application or the Transmission Trigger Timer expiring.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 3-4.12 Transmission Trigger Timer Behavior**

The Transmission Trigger Timer is activated when the Connection transitions to the **Established** state.

**Important:** If the expected_packet_rate attribute contains the value zero (0), then the Transmission Trigger Timer is not activated and/or used by the Client end-point.

### 3-4.5.2 Inactivity/Watchdog Timer

This timer is **required** to be managed by any **consuming** Connection Object. Consuming Connection Objects include:

- Client end-point Connection Objects whose transportClass_trigger attribute indicates either Transport Class 2 or 3
- All Server end-point Connection Objects

This timer is activated when the Connection transitions to the **Established** state. The tasks listed below are performed by a Connection immediately upon detecting that a **valid** message has been consumed:

- The current value for the Inactivity/Watchdog Timer is restored to its initial value and the timer is stopped.
- A new Inactivity/Watchdog Timer is activated.

The bullet items above indicate that the new Inactivity/Watchdog Timer is activated before the received message is processed.

Expiration of this timer is an indication that the Connection Object has timed out while waiting to consume. The Connection Object performs the following steps when the Inactivity/Watchdog timer expires:

- issues an indication of this event to the application
- performs the action indicated by the watchdog_timeout_action attribute


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 3-4.13 Inactivity Watchdog Timer**

Two different values are used for the Inactivity/Watchdog Timer, based on whether or not the Connection Object has consumed a message:

1. The **initial** value loaded into the Inactivity/Watchdog Timer is either 10,000 milliseconds (10 seconds) or the **expected_packet_rate** multiplied by the **connection_timeout_multiplier**, depending on which value is numerically greater[1]. If the **expected_packet_rate** attribute multiplied by the **connection_timeout_multiplier** is greater than 10,000, then the **expected_packet_rate** multiplied by the **connection_timeout_multiplier** is used. Otherwise, 10,000 (10 seconds) is used. This is referred to as the **pre-consumption** timeout value. This value is used because a Connection may transition to the **Established** state before all end-points are fully configured. For example:

1 If the **expected_packet_rate** attribute contains the value zero (0), then the Inactivity/Watchdog Timer is not activated and/or used by the Connection Object. A Connection Object whose **expected_packet_rate** attribute is zero (0) will never experience an Inactivity/Watchdog timeout.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This rule takes into consideration that the application-to-application connection is not really established until the associated information exchange is performed for the first time.

2. All subsequent activations of the Inactivity/Watchdog Timer use the **expected_packet_rate** multiplied by the **connection_timeout_multiplier** as the number of milliseconds to load into the Inactivity/Watchdog Timer [1].

### 3-4.5.3 Production Inhibit Timer

This timer is **required** to be managed by the Connection Object within the **Client end-point** of an I/O Connection when the **production_inhibit_time** attribute value is non-zero. This timer is started only when new data is produced by the Connection Object. Data produced due to the expiration of the Transmission Trigger Timer (see section 3-4.5.1) shall not result in the Production Inhibit Timer being restarted. The Connection Object shall not produce new data if this timer is running. An Acknowledge Handler Object (see section 5A-31.6, Behavior and Configuration of Acknowledged Data Production) retry may, however, be sent to the Link Producer. If one or more new data events occur while this timer is running the Connection Object shall send the most recent new data when this timer expires.

**Figure 3-4.14 Production Inhibit Timer Behavior**

1 If the **expected_packet_rate** attribute contains the value zero (0), then the Inactivity/Watchdog Timer is not activated and/or used by the Connection Object. A Connection Object whose **expected_packet_rate** attribute is zero (0) will never experience an Inactivity/Watchdog timeout.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Important:** The Production Inhibit Timer is initialized with the value in the **production_inhibit_time** attribute. If the **production_inhibit_time** attribute contains the value zero (0), then the Production Inhibit Timer is not activated and/or used by the Client end-point.

**Important:** Server end-points do not activate this timer.

## 3-4.6 Connection Object Instance Services

The Connection Object Instance supports the following CIP Common services:

**Table 3-4.19 Connection Object Instance Services**

| Service Code | Need In Implementation | Service Name | Service Description |
|---|---|---|---|
| 0Ehex | Required | Get_Attribute_ Single | Used to read a Connection Object attribute. |
| 10hex | Optional | Set_Attribute_ Single | Used to modify a Connection Object attribute. The Connection Object returns information in the Service Data Field of a Set_Attribute_Single Response when the **expected_packet_rate** attribute is modified as indicated in Table 3-4.21 |
| 05hex | Optional | Reset | Used to reset the Inactivity/Watchdog Timer associated with a Connection Object. When a Connection in the **Timed Out** or Deferred Delete state receives a Reset request it also transitions back to the **Established** state. |
| 09hex | Optional | Delete | Used to delete a Connection Object and to release all associated resources. |
| 0Dhex | Optional | Apply_Attributes | Used to deliver the Connection Object to the application, which performs the set of tasks necessary to create the specified connection. A Connection Instance returns the **produced_connection_id** and **consumed_connection_id** attributes within the Service Data field of an Apply Attributes Response Message as indicated in Table 3-4.20. The Apply_Attributes service effectively states that the initial configuration of the Connection Object is complete and it is now time to ”activate” that configuration. |

The following information is specified by a Connection Object Instance within the Service Data Field of a successful Apply_Attributes response.

**Table 3-4.20 Service Data for Connection Object Apply Attributes Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Produced Connection ID | UINT | Contains the value within the Connection Instance’s **produced_connection_id** attribute. If the subnet type is DeviceNet, the **DeviceNet_produced_connection_id** attribute is used. |
| Consumed Connection ID | UINT | Contains the value within the Connection Instance’s **consumed_connection_id** attribute. If the subnet type is DeviceNet, the **DeviceNet_consumed_connection_id** attribute is used. |

Figure 3-4.15 below illustrates the Apply_Attributes service sent to a Connection Object Instance. For this example, assume a DeviceNet Explicit Messaging Connection was established across Group 2 and the Message Body Format was defined by the Server as DeviceNet (16/16).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 3-4.15 Apply_Attributes Service Example**

The following information is specified by a Connection Object Instance within the Service Data Field of a successful Set_Attribute_Single response associated with the modification of the **expected_packet_rate** attribute.

**Table 3-4.21 Service Data For Set_Attribute_Single [expected_packet_rate] Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Expected Packet Rate | UINT | Contains the actual value within the Connection Instance's **expected_packet_rate** attribute. |

The illustration in Figure 3-4.16 depicts a Client requesting that the expected_packet_rate be set to 5 milliseconds. The Server returning a successful response stating that the attribute was actually set to 10 milliseconds.

For this example, assume a DeviceNet Explicit Messaging Connection was established across Group 2 and the Message body Format was defined by the Server as DeviceNet (8/8). Assume also that the expected_packet_rate attribute of Connection Instance #1 is requested to be set to 5 milliseconds but the product supports a 10 millisecond resolution.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 3-4.16 Set Expected Packet Rate Example**

- Send_Message - Used internally to trigger the transmission of a message. This results in the invocation of the associated Link Producer's Send service. If the message needs to be transmitted in a fragmented fashion, this service breaks up the message into multiple fragments and invokes the associated Link Producer's Send service multiple times.
- Receive_Data - Used internally to deliver a received frame to the Connection Object. This service is invoked internally by a Link Consumer. The Connection Object is responsible for determining whether or not it must reassemble a series of data fragments into a complete message before processing the message.

## 3-4.7 Connection Instance Behavior

### 3-4.7.1 I/O Connection Instance Behavior

Figure 3-4.17 provides a general overview of the behavior associated with an **I/O Connection Object (instance_type** attribute = I/O).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 3-4.17 I/O Connection Object State Transition Diagram**

A Connection Object issues the internal indications described in Figure 3-4.18 to an application within a product. The purpose of this specification is to describe externally visible behaviors; rules pertinent to specific internal indications are not defined.

**Figure 3-4.18 Internal Indications Issued by a Connection Object (Conceptual)**

**Important:** Table 3-4.22 provides a detailed State Event Matrix for an I/O Connection Object and implementations should be based on this information. This State Event Matrix does not dictate rules with regards to product specific, internal logic. Any attempt to access the Connection Class or a Connection Object Instance may need to pass through product specific verification. This may result in an error scenario that is not indicated by the SEM in Table 3-4.22. This may also result in additional, product specific indications delivered from a Connection Object to the application and/or a specific Application Object. The point to remember is that the Connection Object must exhibit the externally visible behavior specified by the SEM and the attribute definitions (section 3-4.4).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-4.22 I/O Connection State Event Matrix**

| Event | I/O Connection Object State | | | | |
|---|---|---|---|---|---|
| | **Non-Existent** | **Configuring** | **Waiting for Connection ID** | **Established** | **Timed Out** |
| Connection Class receives a Create Request | Class instantiates a Connection Object. Set instance_type to I/O. Set all other attributes to default values. Transition to Configuring | Not applicable | Not applicable | Not applicable | Not applicable |
| Connection Class receives a Delete Request | Error: Object does not exist (General Error Code 16hex) | Release all associated resources. Transition to Non-existent | Release all associated resources. Transition to Non-existent. | Release all associated resources. Transition to Non-existent. | Release all associated resources. Transition to Non-existent. |
| Set_Attribute_ Single | Error: Object does not exist (General Error Code 16hex) | Validate/service the request based on internal logic and per the Access Rules presented in section 3-4.8. Return response. | If request to modify produced or consumed_connection_id then validate the value and service the request. Return appropriate response. If this is a request to access an attribute other than the produced or consumed_connection_id , then return an Error Response whose General Error Code is set to 0Chex (The object cannot perform the requested service in its current mode/state) | Validate/service the request based on internal logic and per the Access Rules presented in section 3-4.8. Return appropriate response. | Validate/service the request based on internal logic and per the Access Rules presented in section 3-4.8. Return appropriate response. |
| Get_Attribute_ Single | Error: Object does not exist (General Error Code 16hex) | Validate/service the request based on internal logic and per the Access Rules presented in section 3-4.8. Return response. | Validate/service the request based on internal logic and per the Access Rules presented in section 3-4.8. Return response. | Validate/service the request based on internal logic and per the Access Rules presented in section 3-4.8. Return response. | Validate/service the request based on internal logic and per the Access Rules presented in section 3-4.8. Return response. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Event | I/O Connection Object State | | | | |
|---|---|---|---|---|---|
| | **Non-Existent** | **Configuring** | **Waiting for Connection ID** | **Established** | **Timed Out** |
| Reset | Error: Object does not exist (General Error Code = 16hex) | Error: The object cannot perform the requested service in its current mode/state. (General Error Code value =0C hex) | Error: The object cannot perform the requested service in its current mode/state. (General Error Code value = 0Chex) | Cancel the current Inactivity/Watchdog Timer. Using the value in the expected_packet_rate attribute, re-start the Inactivity/Watchdog Timer. A success response is returned even if an Inactivity/Watchdog Timer is not utilized by the Connection Object (Client Transport Class 0, expected_packet_rate = 0Chex). | Using the value in the expected_packet_rate attribute, start the Inactivity/Watchdog timer and transition back to the Established state.<br><br>If the expected_packet_rate attribute has been set to zero (0) while the Connection was in the Timed Out state, then just transition back to Established without activating an Inactivity/Watchdog Timer. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Event | I/O Connection Object State | | | | |
|---|---|---|---|---|---|
| | **Non-Existent** | **Configuring** | **Waiting for Connection ID** | **Established** | **Timed Out** |
| Apply_ Attributes | Error: Object does not exist (General Error Code 16hex) | Deliver Connection Object to the application which validates the attribute information. If either of the connection_id attributes (produced or consumed) needs to be configured and cannot be generated by this module, then perform all the other steps necessary to configure the connection, return a Successful Response and transition to the Waiting For Connection ID state. The inability to generate a produced or consumed connection ID value IS NOT reported as an error. If all attributes are validly configured, then perform all the steps necessary to satisfy this connection, start all required timers, and transition to the Established state. If an error is detected, then an Error Response is returned and the Connection remains in the Configuring state[1] and, if a Client Connection with a production trigger value of 0 or 1 (Cyclic or Change of State), produce initial data. | If either of the connection_id attributes (produced or consumed) still needs to be configured and cannot be generated by this module, then return a Successful Response and remain in the Waiting For Connection ID state. The inability to generate a produced or consumed connection ID value IS NOT reported as an error. If the produced and/or consumed_connection_id attributes are now validly configured, then transition to the Established state and return a Successful Response1 and, if a Client Connection with a production trigger value of 0 or 1 (Cyclic or Change of State), produce initial data. | All modifications take place immediately once the Connection has transitioned to the Established state. Return Error: The object cannot perform the requested service in its current mode/state. (General Error Code value = 0Chex) | Error: The object cannot perform the requested service in its current mode/state. (General Error Code value = 0Chex) |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Event | I/O Connection Object State | | | | |
|---|---|---|---|---|---|
| | **Non-Existent** | **Configuring** | **Waiting for Connection ID** | **Established** | **Timed Out** |
| Receive_Data | Not applicable | Discard the message | Discard the message | If a complete, valid [2] message has been received, reset the Inactivity/Watchdog Timer [3] and deliver the I/O Message to the Application. A Connection Object must exhibit the externally visible behavior associated with the current state of its attributes (see section 3-4.4). If this is a fragmented portion of an I/O Message, process as specified by subnet. | Discard the message |
| Send_Message | Not applicable | Return internal error - do not send the message | Return internal error - do not send the message | Transmit the complete I/O Message or fragment as required by the subnet. If this is a Client Connection and the expected_packet_rate attribute is non-zero, restart the Transmission Trigger Timer. | Return internal error - do not send the message |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Event | I/O Connection Object State | | | | |
|---|---|---|---|---|---|
| | **Non-Existent** | **Configuring** | **Waiting for Connection ID** | **Established** | **Timed Out** |
| Inactivity/ Watchdog Timer expires | Not applicable | Not applicable | Not applicable | Examine the watchdog_timeout_action attribute of the Connection and perform the indicated action. If the watchdog_timeout_action attribute indicates that the Connection is to remain in the Established state (*Auto Reset*), then immediately re-start the Inactivity/Watchdog Timer. | Not applicable |

Table Footnotes

1 If the configuration indicates that a Message ID needs to be allocated and an available Message ID does not exist in the specified Message Group, then an Error Response whose General Error Code indicates Resource Unavailable (02hex) is returned. If a Connection Object attribute value passed the range check when it was initially configured but the attribute value conflicts with another piece of information in the node when the Apply request is processed, then an Error Response is returned whose General Error Code is set to Invalid Attribute Value (09hex) and whose Additional Code is set to the Attribute ID of the offending Connection Object Attribute ID.

2 The Connection Object verifies that the length of the received I/O Message is less than or equal to the consumed_connection_size attribute prior to processing the message. If the length of the received message is less than or equal to the consumed_connection_size attribute, then the I/O Connection Object resets the Inactivity/Watchdog Timer, exhibits the externally visible behavior indicated by its attribute settings, and delivers the message to the Application. If the length of the received message is greater than the consumed_connection_size attribute, then the I/O Connection Object immediately discards the message and discontinues any subsequent processing. This is the only message content validation performed by an I/O Connection Object. Subsequent validation must be performed by the Application.

3 If a fragmented message is being received, then the Inactivity/Watchdog Timer is not reset until the entire message has been validly received.

**Important:** The Receive_Data event is only delivered to an I/O Connection when a message whose Connection Identifier Field matches the consumed_connection_id attribute is received. If a message is received whose Connection Identifier Field does not match any Established Connection Object's consumed_connection_id attribute, then the message is discarded. Connection Identifiers are subnet-type specific.

If an implementation detects that it does not support an Explicit Messaging Service indicated in Table 3-4.22, then an Error Response specifying Service Not Supported (General Error Code 08) is returned.

### 3-4.7.2 CIP Bridged Connection Instance Behavior

Figure 3-4.19 provides a general overview of the behavior associated with a **CIP Bridged** Connection **Object** (**instance_type** attribute = CIP Bridged). CIP Bridged connections are used to make connections *offlink*. Both I/O and Explicit Messaging can be accomplished using this connection type. The Connection Manager Object definition provides more details these types of connections.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 3-4.19 CIP Bridged Connection Object State Transition Diagram**

Table 3-4.23 provides a detailed State Event Matrix for a CIP Bridged Connection Object.

**Table 3-4.23 CIP Bridged Connection State Event Matrix**

| Event | CIP Bridged Connection Object State | | | |
|---|---|---|---|---|
| | **Non-Existent** | **Configuring** | **Established** | **Closing** |
| Connection Manager receives a Forward Open Request | Connection Class instantiates a Connection Object. Set **instance_type** to **CIP Bridged**. Set attributes to value delivered by Forward Open service or default for CIP Bridged connections. Transition to **Configuring.** | Not applicable | Not applicable | Not applicable |
| Connection Manager receives notification that connection establishment is complete to target | Not applicable | Transition to **Established** | Not applicable | Not applicable |
| Connection Manager receives a Forward Close Request | Not applicable | Ignore event | Transition to **Closing** | Ignore event |
| Connection Manager receives a Forward Close Response | Not applicable | Release all resources and transition to **Non-Existent** | Release all resources and transition to **Non-Existent** | Release all resources and transition to **Non-Existent** |
| Delete | Error: Object does not exist (General Error Code $16_{hex}$) | Release all resources and transition to **Non-Existent** | Release all resources and transition to **Non-Existent** | Release all resources and transition to **Non-Existent** |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Event | CIP Bridged Connection Object State | | | |
|---|---|---|---|---|
| | **Non-Existent** | **Configuring** | **Established** | **Closing** |
| Get_Attribute_Single | Error: Object does not exist (General Error Code $16_{hex}$) | Validate/service the request based on internal logic and per the Access Rules presented in section 3-4.8. Return appropriate response. | Validate/service the request based on internal logic and per the Access Rules presented in section 3-4.8. Return appropriate response. | Validate/service the request based on internal logic and per the Access Rules presented in section 3-4.8. Return appropriate response. |
| Set_Attribute_Single | Error: Object does not exist (General Error Code $16_{hex}$) | | | |
| Reset | Error: Object does not exist (General Error Code $16_{hex}$) | Error: Service Not Supported (General Error Code $08_{hex}$) | Error: Service Not Supported (General Error Code $08_{hex}$) | Error: Service Not Supported (General Error Code $08_{hex}$) |
| Apply_Attributes | Error: Object does not exist (General Error Code $16_{hex}$) | Error: Service Not Supported (General Error Code $08_{hex}$) | Error: Service Not Supported (General Error Code $08_{hex}$) | Error: Service Not Supported (General Error Code $08_{hex}$) |
| Receive_Data | Not applicable | Ignore event | Invoke send service of connection object on destination port, passing the received data. | Ignore event |
| Send_Message | Not applicable | Ignore event | Send data on subnet. | Ignore event |
| Inactivity/Watchdog Timer expires | Not applicable | Not applicable | Release all resources and transition to **Non-Existent** | Release all resources and transition to **Non-Existent** |

### 3-4.7.3 Explicit Messaging Connection Instance Behavior

Figure 3-4.20 provides a general overview of the behavior associated with an **Explicit Messaging Connection Object** (**instance_type** attribute = Explicit Messaging).

**Figure 3-4.20 Explicit Messaging Connection Object State Transition Diagram**

Table 3-4.24 provides a detailed State Event Matrix for an Explicit Messaging Connection Object. Implementations should be based on the information in Table 3-4.24.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-4.24 Explicit Messaging Connection State Event Matrix**

| Event | Explicit Messaging Connection Object State | | |
|---|---|---|---|
| | **Non-Existent** | **Established** | **Deferred Delete** |
| UCMM receives an Open Explicit Messaging Connection Request/Response and invokes the Create service of the Connection Class | If possible, Class instantiates Connection Object. Set **instance_type** attribute to *Explicit Messaging* [1]. Other attributes are automatically configured using the information in the Open Explicit Messaging Connection Request/Response. Transition to **Established.** If request received, Transmit Open Explicit Messaging Connection Response. | Not applicable | Not applicable |
| UCMM receives a Close Request and invokes the Delete service of the Connection Class | Error: Object does not exist (General Error Code 16hex) | Release all associated resources. Transition to **Non-existent.** | Release all associated resources. Transition to **Non-existent.** |
| Connection Class receives a Delete Request | Error: Object does not exist (General Error Code 16hex) | Release all associated resources. Transition to **Non-existent.** | Release all associated resources. Transition to **Non-existent.** |
| Set_Attribute_Single | Error: Object does not exist (General Error Code 16hex) | Validate/service the request based on internal logic and per the Access Rules presented in section 3-4.8. Return appropriate response. | Validate/service the request based on internal logic and per the Access Rules presented in section 3-4.8. Return appropriate response. |
| Get_Attribute_Single | Error: Object does not exist (General Error Code 16hex) | Validate/service the request based on internal logic and per the Access Rules presented in section 3-4.8. Return appropriate response. | Validate/service the request based on internal logic and per the Access Rules presented in section 3-4.8. Return appropriate response. |
| Reset | Error: Object does not exist (General Error Code 16hex) | Cancel the current Inactivity/Watchdog Timer. Using the value in the **expected_packet_rate** attribute, re-start the Inactivity/Watchdog Timer. | Using the value in the **expected_packet_rate** attribute, re-start the Inactivity/Watchdog Timer **and transition back to the Established state.** |
| Apply_Attributes | Error: Object does not exist (General Error Code 16hex) | Error: The object cannot perform the requested service in its current mode/state. (General Error Code value = 0Chex) | Error: The object cannot perform the requested service in its current mode/state. (General Error Code value = 0Chex) |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Event | Explicit Messaging Connection Object State | | |
|---|---|---|---|
| | **Non-Existent** | **Established** | **Deferred Delete** |
| Receive_Data | Not applicable | If a valid message or message fragment has been received, then reset the Inactivity/Watchdog Timer[2]. Either process/store the fragment or handle the Explicit Message. | If a valid message or message fragment has been received, then restart the Inactivity/Watchdog Timer[2] and transition back to the **Established** state. Either process/store the fragment or handle the Explicit Message. |
| Send_Message | Not applicable | Examine the length of the message to transmit and, if necessary, perform a fragmented series of transmissions. Otherwise, transmit the complete Explicit Message. | Examine the length of the message to transmit and, if necessary, perform a fragmented series of transmissions. Otherwise, transmit the complete Explicit Message. |
| Inactivity/Watchdog Timer expires | Not applicable | If the **watchdog_timeout_action** attribute is set to *Auto Delete* or is set to *Deferred Delete* and no child connection instances are in the **Established** state release all associated resources and Transition to **Non-existent**. If the **watchdog_timeout_action** attribute is set to *Deferred Delete* and at least one child connection instance is in the **Established** state transition to **Deferred Delete** | Not applicable. |
| Child connection instance Deleted or Transitions to Timed Out | Not applicable | Ignore event | If no other child connection instances are in the **Established** state release all associated resources and transition to **Non-existent.** |

Table Footnotes:

1 If the configuration indicates that a connection resource needs to be allocated and an available resource does not exist, then an Error Response who's General Error Code indicates Resource Unavailable (02hex) is returned.

2 On DeviceNet, the MAC ID field within the Message Header of all Explicit Messages and/or Message Fragments is examined. If the Destination MAC ID is specified in the Connection ID (CAN Identifier Field), then the Source MAC ID of the other end-point must be specified in the Message Header. If the Source MAC ID is specified in the Connection ID, then the receiving module's MAC ID must be specified in the Message Header. If either of these checks fail, then the Inactivity/Watchdog Timer IS NOT reset and the message/message fragment is discarded.

**Important:** The Receive_Data event is only delivered to an Explicit Messaging Connection when a message who's Connection ID matches the consumed_connection_id attribute is received. If a message is received whose connection id does not match any Established Connection Object's consumed_connection_id attribute, then the message is discarded.

If an implementation detects that it does not support an Explicit Messaging Service indicated in Table 3-4.24, then an Error Response specifying Service Not Supported (General Status Code 08) is returned.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 3-4.8 Connection Object Attribute Access Rules

During the configuration of a Connection instance using the Set_Attribute service, a module must perform value checks of each separate attribute when is modified. If an error is detected, then an Error Response is returned. The discovery of an error DOES NOT cause the deletion of the Connection instance.

**Important:** If the produced_connection_id and/or consumed_connection_id attributes contain a non-default value upon reception of the Apply Request, then the related portion of the initial_comm_characteristics attribute is ignored and the ID value is validated and used.

The following series of tables indicates when an attribute can be read or written via a Get and/or Set operation based on the Connection's **state** and **instance_type.**

**Important:** If a request to Get/Set a supported attribute is received but the current state and/or instance_type of Connection dictates that the requested access is invalid, then the returned error status indicates: *The object cannot perform the requested service in its current mode/state* (General Status Code value = $0C_{hex}$ - see Appendix B for more information on status codes).

**Table 3-4.25 I/O Connection Object Attribute Access**

| Attribute | I/O Connection State | | | | |
|---|---|---|---|---|---|
| | **Non-Existent** | **Configuring** | **Waiting for Connection ID** | **Established** | **Timed Out** |
| State | Not available | Get Only | Get Only | Get Only | Get Only |
| instance_type | Not available | Get Only | Get Only | Get Only | Get Only |
| transport Class_trigger | Not available | Get/Set | Get Only | Get Only | Get Only |
| produced_connection_id | Not available | Get/Set | Get/Set | Get/Set | Get/Set |
| consumed_connection_id | Not available | Get/Set | Get/Set | Get/Set | Get/Set |
| initial_comm_characteristics | Not available | Get/Set | Get Only | Get Only | Get Only |
| produced_connection_size | Not available | Get/Set | Get Only | Get Only | Get Only |
| consumed_connection_size | Not available | Get/Set | Get Only | Get Only | Get Only |
| expected_packet_rate[1] | Not available | Get/Set | Get Only | Get/Set | Get/Set |
| watchdog_timeout_action | Not available | Get/Set | Get Only | Get/Set | Get/Set |
| produced_connection_ path_length | Not available | Get Only | Get Only | Get Only | Get Only |
| produced_connection_path | Not available | Get/Set | Get Only | Get Only | Get Only |
| consumed_connection_ path_length | Not available | Get Only | Get Only | Get Only | Get Only |
| consumed_connection_path | Not available | Get/Set | Get Only | Get Only | Get Only |
| production_inhibit_time | Not available | Get/Set | Get Only | Get/Set | Get/Set |
| connection_timeout_multiplier | Not available | Get/Set | Get Only | Get/Set [1] | Get/Set |
| Connection_binding_list | Not available | Get Only | Get Only | Get Only | Get Only |

Table Footnotes:

1 When a Connection Object is in the **Established** state, any modifications to the **expected_packet_rate** or **connection_timeout_multiplier** attributes have immediate effect on the Inactivity/Watchdog Timer. The following steps are performed by a Connection Object in the **Established** state when a request is received to modify the **expected_packet_rate or connection_timeout_multiplier** attribute:

- the current Inactivity/Watchdog Timer is canceled
- a new Inactivity/Watchdog Timer is activated based on the new value in the expected_packet_rate or connection_timeout_multiplier attribute.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-4.26 CIP Bridged Connection Object Attribute Access**

| Attribute | Default Value[1] | CIP Bridged Connection State | | | |
|---|---|---|---|---|---|
| | | Non-Existent | Configuring | Established | Closing |
| state | 1 | Not Available | Get Only | Get Only | Get Only |
| instance_type | 2 | Not Available | Get Only | Get Only | Get Only |
| transportClass_trigger | From service | Not Available | Get Only | Get Only | Get Only |
| produced_connection_id | From service | Not Available | Get Only | Get Only | Get Only |
| consumed_connection_id | From service | Not Available | Get Only | Get Only | Get Only |
| initial_comm_characteristics | From service | Not Available | Get Only | Get Only | Get Only |
| produced_connection_size | From service | Not Available | Get Only | Get Only | Get Only |
| consumed_connection_size | From service | Not Available | Get Only | Get Only | Get Only |
| expected_packet_rate | From service | Not Available | Get Only | Get Only | Get Only |
| watchdog_timeout_action | 1 | Not Available | Get Only | Get Only | Get Only |
| produced_connection_path_length | From service | Not Available | Get Only | Get Only | Get Only |
| produced_connection_path | From service | Not Available | Get Only | Get Only | Get Only |
| consumed_connection_path_length | From service | Not Available | Get Only | Get Only | Get Only |
| consumed_connection_path | From service | Not Available | Get Only | Get Only | Get Only |
| production_inhibit_time | From service | Not Available | Get Only | Get Only | Get Only |
| Connection_timeout_multiplier | From service | Not Available | Get Only | Get Only | Get Only |
| Connection_binding_list | Empty | Not Available | Get Only | Get Only | Get Only |

Table Footnotes:

1 The default value is either the value indicated or is set based on one of the parameters of the Forward Open service received by the Connection Manager object.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-4.27 Explicit Messaging Connection Object Attribute Access**

| Attribute | Explicit Messaging Connection State | |
|---|---|---|
| | **Non-Existent** | **Established/Deferred Delete** |
| State | Not available | Get Only |
| instance_type | Not available | Get Only |
| transportClass_trigger | Not available | Get Only |
| produced_connection_id | Not available | Get Only |
| consumed_connection_id | Not available | Get Only |
| initial_comm_characteristics | Not available | Get Only |
| produced_connection_size | Not available | Get Only |
| consumed_connection_size | Not available | Get Only |
| expected_packet_rate[1] | Not available | Get/Set[1] |
| watchdog_timeout_action | Not available | Get/Set |
| produced_connection_ path_length | Not available | Get Only |
| produced_connection_path | Not available | Get Only |
| consumed_connection_ path_length | Not available | Get Only |
| consumed_connection_path | Not available | Get Only |
| production_inhibit_time | Not available | Get Only |
| connection_timeout_multiplier | Not available | Get/Set [1] |
| Connection_binding_list | Not available | Get Only |

Table Footnotes:

1 When a Connection Object is in the Established state, any modifications to the expected_packet_rate or connection_timeout_multiplier attributes have immediate effect on the Inactivity/Watchdog Timer. The following steps are performed by a Connection Object in the Established state when a request is received to modify the expected_packet_rate or connection_timeout_multiplier attribute:

- the current Inactivity/Watchdog Timer is canceled
- a new Inactivity/Watchdog Timer is activated based on the new value in the expected_packet_rate or connection_timeout_multiplier attribute.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 3-5 Connection Manager Object Class Definition

**Class Code: 06 hex**

The Connection Manager Class allocates and manages the internal resources associated with both I/O and Explicit Messaging Connections. The specific instance generated by the Connection Manager Class is referred to as a Connection Instance or a Connection Object.

## 3-5.1 Class Attributes

The Connection Manager Class attributes are defined below in Table 3-5.1.

**Table 3-5.1 Connection Manager Class Attributes**

| Attribute ID | Need In Implementation | Access Rule | Attribute Name | Data Type | Attribute Description | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |

## 3-5.2 Instance Attributes

The following list provides a summary of the Connection Manager Instance attributes and their associated data types.

**Table 3-5.2 Connection Manager Object Instance Attributes**

| Attr ID | Need In Implem | Access Rule | N V | Attribute Name | Data Type | Description of Attribute | Semantics |
|---|---|---|---|---|---|---|---|
| 1 | Optional | Set [1] | V | Open Requests | UINT | Number of Forward Open service requests received. | |
| 2 | Optional | Set [1] | V | Open Format Rejects | UINT | Number of Forward Open service requests which were rejected due to bad format. | |
| 3 | Optional | Set [1] | V | Open Resource Rejects | UINT | Number of Forward Open service requests which were rejected due to lack of resources. | |
| 4 | Optional | Set [1] | V | Open Other Rejects | UINT | Number of Forward Open service requests which were rejected for reasons other than bad format or lack of resources. | |
| 5 | Optional | Set [1] | V | Close Requests | UINT | Number of Forward Close service requests received. | |
| 6 | Optional | Set [1] | V | Close Format Rejects | UINT | Number of Forward Close service requests which were rejected due to bad format. | |
| 7 | Optional | Set [1] | V | Close Other Rejects | UINT | Number of Forward Close service requests which were rejected for reasons other than bad format. | |
| 8 | Optional | Set [1] | V | Connection Timeouts | UINT | Total number of connection timeouts that have occurred in connections controlled by this Connection Manager | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need In Implem | Access Rule | N V | Attribute Name | Data Type | Description of Attribute | Semantics |
|---|---|---|---|---|---|---|---|
| 9 | Optional | Get | V | Connection Entry List | STRUCT of | List of Connections | If the Connection Object (Class Code 0x05) is supported, the index value into the ConnOpenBits array is the Connection Instance number minus one (the first entry in the array, index value 0, is for Connection Instance 1). |
| | | | | NumConnEntries | UINT | Number of connection entries. This attribute, divided by 8 and rounded up for any remainder, gives the length of the array (in bytes) of the ConnOpenBits field of this structure. | Number of bits in the ConnOpenBits attribute. |
| | | | | ConnOpenBits | ARRAY of BOOL | List of connection data which may be individually queried by the Get/Search Connection Data Services. Each bit represents a possible connection. | 0 = Connection Instance is Non-Existent.<br>1 = Connection Instance is Existent. Query for more information. |
| 10 | Reserved/Obsolete | | | | | | |
| 11 | Optional | Get | V | CPU_Utilization | UINT | CPU Utilization in tenths of a percent. | See section 3-5.2.1.1 |
| 12 | Optional | Get | V | MaxBuffSize [2] | UDINT | Amount of buffer space originally available. | Size in bytes |
| 13 | Optional | Get | V | BufSize Remaining [2] | UDINT | Amount of buffer space available at this time. | Size in bytes |
| 14 | Optional | Get | N V | Max Connection Establishment Time | UINT | Number of milliseconds<br>1 to 65,535 | See section 3-5.2.1.2 |
| 15 | Optional | Get | V | I/O Packets per Second | UDINT | Represents the current I/O packets per second for I/O traffic | See semantics in section 3-5.2.1.3 |
| 16 | Optional | Get | V | Percent I/O Utilization | UINT | Represents the current I/O communications utilization for I/O traffic | See semantics in section 3-5.2.1.4 |
| 17 | Optional | Get | V | Explicit Packets Per Second | UDINT | Total number of explicit packets sent and received by this device over the last second. | See semantics in section 3-5.2.1.5 |
| 18 | Conditional[3] | Set[1] | V | Missed I/O Packets | UDINT | A running total of missed I/O packets. | See semantics in section 3-5.2.1.6 |
| 19 | Optional | Get | V | CIP I/O Connections | UDINT | Total number of CIP I/O connections in use | See semantics in section 3-5.2.1.7 |
| 20 | Optional | Get | V | CIP Explicit Connections | UDINT | Total number of CIP Explicit Messaging connections in use | See semantics in section 3-5.2.1.8 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need In Implem | Access Rule | N V | Attribute Name | Data Type | Description of Attribute | Semantics |
|---|---|---|---|---|---|---|---|

Table Footnotes:
1. A device may reject a set request to this attribute, using General Status Code 0x09 (Invalid Attribute Value), if the attribute value sent is not zero.
2. The meaning of, and method of calculating, these values are vendor specific.
3. This attribute is Optional for devices that have an EtherNet/IP interface. Devices with no EtherNet/IP interface are not permitted to implement this attribute.

### 3-5.2.1 Semantics

#### 3-5.2.1.1 Percent CPU Utilization (Attribute 11)

The Percent CPU Utilization attribute indicates the percentage of capacity of the product's compute engine (whether that be a CPU and/or a core of a CPU) that is most important to the performance of the product. Due to the variations in architectures of various products, each product needs to define this value based on its design, hence the method of calculating this value is vendor specific.

This value is intended to indicate the current loading of the device and may be approaching a point of saturation and performance degradation. This is not simply a measure of how many packets per second are flowing to/from this device, though this is typically a factor in the calculation. It should factor in all CPU processing resources used by the device.

In products that support an operating system, this is often provided by the operating system as the inverse of the time the idle task runs, but that may or may not be a good indication of what's really left for communications related functions. For devices where this is not provided by the OS already, or there's no OS used, the vendor must determine which internal resources should be monitored to determine what can indicate when a device is reaching saturation, and then turn that into a percentage of the max.

Here are examples of what the percentages mean:

1. 95% - 100% – Device is overloaded
2. 80% - 95% - Device is approaching a point of saturation and/or performance degradation
3. Below 80% - Device is OK

Vendors shall put in their documentation factors that will cause the device to go above 80%.

This UINT has a range of 0-1000, representing 0.0% to 100.0%.

#### 3-5.2.1.2 Max Connection Establishment Time (Attribute 14)

The Max Connection Establishment Time is the worst case Forward Open response time for any connection path of the device. It is recommended that devices whose Max Connection Establishment Time is greater than four seconds support this attribute. See section 3-5.4.1.3 Unconnected Request Timing for more information.

When this attribute is implemented, a corresponding entry keyword shall be included in the EDS file of the device, see section 7-3.6.10 Connection Manager Section and/or section 7-3.6.11 Port Specific Connection Manager Section.

#### 3-5.2.1.3 I/O Packets per Second (Attribute 15)

The I/O Packets per Second attribute indicates the total number of I/O (implicit) packets produced and consumed by this device in the previous second, including duplicate packets.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 3-5.2.1.4 Percent I/O Utilization (Attribute 16)

The Percent I/O Utilization attribute indicates what percentage of the I/O (implicit) communications resources are in use in this device. This value is calculated based on the device's Traffic Specs for various packet sizes. Refer to the description of the Traffic Spec in Chapter 7-3.6.13 for more details.

This UINT is in units of 0.1%, with 1100 representing 110.0% utilization.

#### 3-5.2.1.5 Explicit Packets Per Second (Attribute 17)

The Explicit Packets per Second attribute indicates the total number of explicit (unconnected and connected) packet requests and replies sent and received by this device in the previous second, including duplicate packets.

#### 3-5.2.1.6 Missed I/O Packets (Attribute 18)

For each connection, as long as the connection remains open, the consumer will subtract the previously received Encapsulation Sequence Number from the current one. If the value exceeds positive one (+1), then that value minus one is the number of packets that have been missed. A connection timeout does not cause Missed Packets to increment.

#### 3-5.2.1.7 CIP I/O Connections (Attribute 19)

The CIP I/O Connections attribute contains the count of open CIP I/O (implicit) connections. This value includes connections that have an endpoint in this device, as well as routed connections that go through this device and out any other port.

#### 3-5.2.1.8 CIP Explicit Connections (Attribute 20)

The CIP Explicit Connections attribute contains the count of the open CIP Explicit Messaging connections (Class 3 to Message Router). This value includes connections that have an endpoint in this device, as well as routed connections that go through this device and out any other port.

### 3-5.3 Common Services

The Connection Manager Object supports the following Common services:

**Table 3-5.3 Connection Manager Object Common Services**

| Service Code | Need In Implementation | | Service Name | Service Description |
|---|---|---|---|---|
| | Class | Instance | | |
| 0x01 | Optional | Optional | Get_Attributes_All | Returns the contents of all attributes |
| 0x02 | n/a | Optional | Set_Attributes_All | Modifies the value of a list of attributes (see the Set_Attributes_All Request definition below) |
| 0x0E | Conditional [1] | Conditional [1] | Get_Attribute_Single | Used to read a Connection Manager Object attribute. |
| 0x10 | n/a | Optional | Set_Attribute_Single | Used to modify a Connection Manager Object attribute. |

Table Footnotes:

1. This service is Required if any of the Connection Manager attributes are supported and the Get_Attributes_All service is not supported.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-5.3.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 3-5.4 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 |
| 2 | UINT | Max Instance | 1 |
| 6 | UINT | Maximum ID Number Class Attributes | 0 |
| 7 | UINT | Maximum ID Number Instance Attributes | 0 |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level,** the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows. This service shall only return values of supported attributes. Therefore, a device can only return values up to the last consecutive supported attribute starting from the first attribute (Attribute 1). Values from any attributes supported beyond these must be acquired via a Get_Attribute_Single service.

**Table 3-5.5 Get_Attributes_All Response Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented)[2] |
|---|---|---|---|
| 1 | UINT | Open Requests | |
| 2 | UINT | Open Format Rejects | |
| 3 | UINT | Open Resource Rejects | |
| 4 | UINT | Open Other Rejects | |
| 5 | UINT | Close Requests | |
| 6 | UINT | Close Format Rejects | |
| 7 | UINT | Close Other Rejects | |
| 8 | UINT | Connection Timeouts | |
| 9 | STRUCT of: | Connection Entry List | |
| | UINT | NumConnEntries[1] | |
| | ARRAY of BOOL | ConnOpenBits | |
| 11 | UINT | CPU_Utilization | |
| 12 | UDINT | MaxBuffSize | |
| 13 | UDINT | BufSize Remaining | |
| 14 | UINT | Max Connection Establishment Time | |

Table Footnotes:

1. This field, divided by 8 and rounded up for any remainder, gives the length of the array (in bytes) of the ConnOpenBits field of this structure.
2. No default values are provided because no unsupported attributes are allowed as described above.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-5.3.2 Set_Attributes_All Request

At the Instance Level, the order of the attributes sent in the Attribute Values portion of the Set_Attributes_All request is as follows:

**Table 3-5.6 Set_Attributes_All Request Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Values (if not implemented) |
|---|---|---|---|
| 1 | UINT | Open Requests[1] | 0 |
| 2 | UINT | Open Format Rejects[1] | 0 |
| 3 | UINT | Open Resource Rejects[1] | 0 |
| 4 | UINT | Open Other Rejects[1] | 0 |
| 5 | UINT | Close Requests[1] | 0 |
| 6 | UINT | Close Format Rejects[1] | 0 |
| 7 | UINT | Close Other Rejects[1] | 0 |
| 8 | UINT | Connection Timeouts[1] | 0 |

Table Footnotes:

1. If this attribute is not supported, the value sent shall be ignored. This condition does not cause the service request to fail.

## 3-5.4 Object-Specific Services

The Connection Manager Object supports the following Object-specific services:

**Table 3-5.7 Connection Manager Object Object-Specific Services**

| Service Code | Need In Implementation | | Service Name | Service Description |
|---|---|---|---|---|
| | Class | Instance | | |
| 0x4E | n/a | Conditional | Forward_Close | Closes a connection |
| 0x52 | n/a | Conditional | Unconnected_Send | Unconnected Send Service. Only originating devices and devices that route between links need to implement |
| 0x54 | n/a | Conditional | Forward_Open | Opens a connection, maximum data size is 511 bytes |
| 0x56 | n/a | Optional | Get_Connection_Data | For diagnostics of a connection |
| 0x57 | n/a | Optional | Search_Connection_Data | For diagnostics of a connection |
| 0x59 | n/a | Obsolete | | |
| 0x5A | n/a | Conditional [1] | Get_Connection_Owner | Determine the owner of a redundant connection |
| 0x5B | n/a | Optional | Large_Forward_Open | Opens a connection, maximum data size is 65535 bytes |

Table Footnotes:

1. Shall be implemented in any device that accepts redundant connections

### 3-5.4.1 Connection Manager Object Specific Service Parameters

The object specific services of the Connection Manager Object share many of the same service parameters. These parameters are defined in this section and referenced in the object specific service definitions. The parse order of the parameters is device specific, although checking the Electronic Key segment if present in the Connection Path first is highly recommended.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The Forward_Open, Large_Forward_Open and Forward_Close services shall be sent using the UCMM or an unbridged (local) explicit messaging connection only. They shall not be sent over a bridged explicit messaging connection. This restriction is required since each node on a bridged connection needs to receive and process these services. On subnets which support UCMM messaging, the recipient (either a target or a router) shall support these services using the UCMM and, in addition, may support these services over a local explicit messaging connection. On subnets which do not support UCMM messaging, these services shall be sent using a local explicit messaging connection. See Chapter 10 for more details on bridging and routing.

#### 3-5.4.1.1 Network Connection Parameters

There are two sets of Network Connection Parameters in the Forward_Open and the Large_Forward_Open: one for the O→T network connection and one for the T→O network connection.

Each set of Network Connection Parameters in the Forward_Open shall be provided as a single 16-bit word that contains the fields in the following figure:

**Table 3-5.8 Network Connection Parameters for Forward_Open**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Redundant Owner | Connection Type | | Reserved | Priority | | Fixed / Variable | Connection Size (in bytes) | | | | | | | | |

Each set of Network Connection Parameters in the Large_Forward_Open shall be provided as a single 32-bit word that contains the fields in the following figure:

**Table 3-5.9 Network Connection Parameters for Large_Forward_Open**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24-16 | 15-0 |
|---|---|---|---|---|---|---|---|---|
| Redundant Owner | Connection Type | | Reserved | Priority | | Fixed / Variable | Reserved | Connection Size (in bytes) |

- *Connection Size:* The maximum size, in bytes, of the data for each direction (where applicable) of the connection. For a variable sized connection, the size shall be the maximum size of the buffer for any transfer. The actual size of the transfer for a variable connection shall be equal to or less than the size specified for the network connection. The maximum buffer size shall be dependent on the links that the connection traverses. The connection size includes the sequence count and the 32-bit real time header, if present.
- *Fixed / Variable*: With a fixed size connection, the amount of data on each transmission shall be the size specified in the *Connection Size* parameter. With a variable size connection, the amount of data on each transmission may be a variable size, up to the size specified in the *Connection Size* parameter.
  0 = Fixed
  1 = Variable
- *Priority*: The priority shall be one of (see Section 3-5.4.1.1.1, Priority for definition and usage):
  00 = Low
  01 = High
  10 = Scheduled
  11 = Urgent


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- *Connection Type* (see Section 3-5.4.1.1.2, Connection Type for definition and usage):
  00 = Null (may be used to reconfigure a connection)
  01 = Multicast
  10 = Point to Point
  11 = Reserved
- *Redundant Owner*
  The redundant owner bit in the O⇒T direction shall be set (= 1) to indicate that more than one owner may be permitted to make a connection simultaneously. The bit shall be clear (= 0) to indicate an exclusive-owner, input only or listen-only connection.
- *Reserved* fields shall be set to zero

#### 3-5.4.1.1.1 Priority

The order of priorities, from highest to lowest are: Urgent, Scheduled, High and Low.

While there are no specific implementation requirements for end devices to internally differentiate traffic with different priorities, devices are recommended to give preferential processing to higher priority connections over lower priority connections. At a minimum, devices are recommended to give higher priority to processing implicit messages over explicit messages.

The network specific volumes define the mapping of CIP Priorities to link priorities.

***Urgent priority***

Urgent priority shall only be used where defined explicitly in the CIP specification volumes, for example CIP Motion I/O connections.

On ControlNet:

- Bandwidth shall be reserved using the Network Schedule segment
- The data interval shall be restricted to the API, which means that if data arrives at a router faster than the API, the node shall filter the packets to the API on ControlNet.
- A tool that performs scheduling operations shall favor urgent priority over scheduled priority connections.

On other CIP links:

- Nodes shall map the Urgent priority to the corresponding network specific link priority (see network specific volume).

***Scheduled priority***

On ControlNet:

- Bandwidth shall be reserved using the Network Schedule segment
- The data interval shall be restricted to the API, which means that if data arrives at a router faster than the API, the node shall filter the packets to the API on ControlNet.
- A tool that performs scheduling operations shall favor urgent priority over scheduled priority connections.

On other CIP links:

- Nodes shall map the Scheduled priority to the corresponding network specific link priority (see network specific volume).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

***High priority***

Nodes shall map the High priority to the corresponding network specific link priority (see network specific volume).

***Low priority***

Nodes shall map the Low priority to the corresponding network specific link priority (see network specific volume).

#### 3-5.4.1.1.2 Connection Type

The Connection Type can be set to one of the following types: Null, Multicast, or Point to Point.

***Null***

This Connection Type does not open up a network connection. For uses of the Null Connection Type, see 3-5.4.2 (Forward_Open and Large_Forward_Open Request).

***Multicast***

This Connection Type shall open a network connection where the data is sent using a destination address that can be received by multiple consumers. The same underlying network connection shall be reused by subsequent multicast connections that specify the same application path. For uses of the Multicast Connection Type, see 3-6.3 (T→O Network Connection Rules).

***Point to Point***

This Connection Type shall open a network connection where the data is sent only between one producer and one consumer.

#### 3-5.4.1.2 Requested and Actual Packet Intervals

The connection originator provides a requested time between productions for each direction of the connection. The target provides the actual time between productions. When a connection traverses ControlNet, the actual time may be further adjusted as provided in Section 3-5.4.1.1.1, Priority.

The requested packet interval (RPI) shall be the requested time between packets in microseconds. The format of the RPI shall be a 32-bit integer in microseconds.

The actual packet interval (API) shall be the maximum time between packets in microseconds. The API is equivalent to the Transmission Trigger Timer value described in section 3-4.5.1. The format of the API shall be a 32-bit integer in microseconds.

The influence of the API and RPI values on the behavior of the connection is dependent on the Transport Class and Trigger definitions in section 3-4.4.3. The RPI value shall be used to allocate capacity usage at each of the producing nodes. The allocation of capacity usage may have to be adjusted when the API is returned, since it is possible for the RPI and API to differ.

The API shall equal the RPI within the device's timer resolution limits, except as indicated for ControlNet in Section 3-5.4.1.1.1, Priority. If the API is not equal to the RPI it shall be faster than the RPI unless the resulting API would be 0, in which case the RPI can't be supported and an error shall be returned, either:

- RPI Value(s) Not Acceptable, General Status 0x01, Extended Status 0x0112 (recommended).
- RPI Not Supported, General Status 0x01, Extended Status 0x0111.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The Inactivity/Watchdog time-out value at each of the intermediate and target nodes shall be set to the connection time-out multiplier times the RPI. If the resulting time-out is not achievable within the device's timer resolution limits, the time-out shall be rounded up the next value within the device's timer resolution limits.

#### 3-5.4.1.2.1 Unconnected Request Timing

The Priority/Time_tick parameter determines the priority of the unconnected message and the time duration of a 'tick' (Tick Time) specified in the Time-out_ticks parameter. The bit fields are:

**Table 3-5.10 Priority/Time_tick Bit Definition**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | Priority<br>0 = Normal<br>1 = Reserved | Tick Time | | | |

The Reserved and Priority fields shall be set to zero (0). The tick time shall be used in conjunction with the Time-out_ticks parameter value to determine the total time out value. The following formula is used:

$$\text{Actual Time Out value} = 2^{\text{time_tick}} \times \text{Time_out_tick}$$

If the tick time is 0000 (1 ms.), a Time-out_ticks value of 5 would translate into 5 ms. If the tick time is 0010 (4 ms.), a Time-out_ticks value of 5 would translate into 20 ms. The Tick Time value is enumerated in the following table.

**Table 3-5.11 Time Tick Value Enumeration**

| Time Tick Value Enumeration | | |
|---|---|---|
| **Tick Time (binary)** | **Time per Tick** | **Max Time** |
| 0000 | 1 ms | 255 ms |
| 0001 | 2 | 510 |
| 0010 | 4 | 1020 |
| 0011 | 8 | 2040 |
| 0100 | 16 | 4080 |
| 0101 | 32 | 8160 |
| 0110 | 64 | 16,320 |
| 0111 | 128 | 32,640 |
| 1000 | 256 | 65,280 |
| 1001 | 512 | 130,560 |
| 1010 | 1024 | 261,120 |
| 1011 | 2048 | 522,240 |
| 1100 | 4096 | 1,044,480 |
| 1101 | 8192 | 2,088,960 |
| 1110 | 16,384 | 4,177,920 |
| 1111 | 32, 768 | 8, 355, 840 ms |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

***Time-out_ticks***

The Time-out ticks parameter shall be used to specify the amount of time the originating application shall wait for the transaction to be completed. When used with the Time Tick portion of the Priority/Time_tick field, a total timeout value can be calculated.

***Timeout Algorithm***

The *Priority/Time_tick* and *Time-out ticks* parameters are used to convey timeout information as an unconnected request flows through interconnecting devices. The originator states how long it will wait for a response (total round trip time).

Each router subtracts the approximate amount of time it needs to process and forward both the request and the associated response. If a router cannot determine specifically how much time to subtract, it shall subtract 512 milliseconds.

When encoding the result of the subtraction, the router shall use the smallest *Time_Tick* possible (see Table 3-5.11) and ensure that the outgoing value is less than the received value. This provides the best granularity for decreasing the timeout as the message traverses the system. This also means that the *Time_Tick* may change as the message is passed through multiple hops.

Each router shall use the new value as a timer for the resources used to manage the unconnected message and also to check to see if a timeout is imminent, i.e. the remaining time is zero or less, before forwarding the message. If a timeout is imminent, the router returns a General Status code of 0x01 with Extended Status code of 0x204 (UNCONNECTED REQUEST TIMED OUT) error response immediately rather than just letting the originator timeout.

If a router can detect that the next hop device is not present (e.g. the router supports an active node table function), it returns a General Status code of 0x01 with Extended Status code of 0x0204 (UNCONNECTED REQUEST TIMED OUT) error response immediately.

When a router fails to receive a response in the remaining time, a General Status code of 0x01 with Extended Status code of 0x0204 (UNCONNECTED REQUEST TIMED OUT) error response is returned.

When a target receives a Forward_Open, Large_Forward_Open or Forward_Close request, the target may either:

1. ignore the *Priority/Time_tick* and *Time-out ticks* parameters and process the request or
2. evaluate the *Priority/Time_tick* and *Time-out ticks* parameters and return a General Status code of 0x01 with Extended Status code of 0x0204 (UNCONNECTED REQUEST TIMED OUT) error response immediately if it knows it will not have enough time to complete the request.

If a target evaluates the *Priority/Time_tick* and *Time-out ticks* parameters:

- Adding the timing evaluation to a target as part of a product update needs to be carefully considered. An updated product shall use its **best** case time for comparison. Thus, an updated target should always accept connections that previously succeeded.
- Supporting the timing evaluation in a target as part of a new product shall use its **worst** case time for comparison. The result is that the target (if present) sends the timeout response instead of any router device.
- If a new or updated product can emulate one or more existing device(s), it shall perform the timing behavior of the device indicated in the compatible key of the request.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

In all error cases, the node that generates the error response includes the Remaining Path Size parameter and every node that receives it releases all resources associated with that unconnected message. The Remaining Path Size in the error response facilitates the ability to know how far along the route the request progressed before encountering the error

When an UNCONNECTED REQUEST TIMED OUT error is encountered, it is most likely due to one of the devices in the path (including the target) not being available (e.g. powered off, cable disconnected, etc.). Though less likely, the UNCONNECTED REQUEST TIMED OUT error may also result when there is network congestion and/or one or more of the devices along the path including the target take longer than the time provided by the originator to process the unconnected request. When an UNCONNECTED REQUEST TIMED OUT error persists and it is known through other means that all nodes are available, the originator timeout value should be increased after taking the topology into account.

A default originator timeout value of 5 seconds is recommended; based on the case of an originator request covering 2 hops (allowing the default 512 milliseconds for each hop and about 4 seconds for the target).

#### 3-5.4.1.3 Connection Serial Number

The connection serial number shall be a unique 16-bit value selected by the connection manager at the originator of the connection. The originator shall make sure that the 16-bit value is unique for the device. There shall be no other significance placed on the number by any other nodes in the connection path. The connection serial numbers shall be unique but do not have to be sequential. For example, an operator interface may have a large number of connections open at the same time, each with a unique number. The same values could be repeated at other operator interface stations. A possible implementation would be to have a connection list which points to the descriptor for each connection, and the connection serial number could be the index into the table.

#### 3-5.4.1.4 Connection Timeout Multiplier

The Connection Timeout Multiplier specifies the multiplier applied to the RPI to obtain the connection timeout value. Devices shall stop transmitting on a connection whenever the connection times out even if the pending close has been sent. The multiplier shall be as represented by the following table:

**Table 3-5.12 Connection Timeout Multiplier Values**

| Value | Multiplier |
|---|---|
| 0 | x4 |
| 1 | x8 |
| 2 | x16 |
| 3 | x32 |
| 4 | x64 |
| 5 | x128 |
| 6 | x256 |
| 7 | x512 |
| 8 - 255 | Reserved |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 3-5.4.1.5 Originator Vendor ID

The Originator Vendor ID utilized in conjunction with the Connection Manager is a reference to the Identity object instance #1, attribute #1 (Vendor ID) of the connection originator.

#### 3-5.4.1.6 Originator Serial Number

The Originator Serial Number utilized in conjunction with the Connection Manager is a reference to the Identity object instance #1, attribute #6 (Serial Number) of the connection originator.

#### 3-5.4.1.7 Connection Triad

The Connection Triad used in the Connection Manager specification relates to the combination of Connection Serial Number, Originator Vendor ID and Originator Serial Number parameters.

#### 3-5.4.1.8 Connection Number

The connection number shall be a 16-bit value that is assigned by the connection manager when a connection is opened. This value allows other nodes to obtain connection data from the connection manager. This number shall not be confused with the Connection Serial Number.

#### 3-5.4.1.9 Connection Path Size

The connection path size shall be the length of the connection path in 16-bit words. The length of the connection path varies during the connection process, since each node in the connection path removes the current port segment and forwards only the remaining path segments to the next node.

Appendix C, sections C-1.3 and C-1.4 specify the components of a connection path and sections C-1.5 and C-1.6 specify the hierarchy of a connection path.

When a configuration data segment is included in the connection path the configuration data segment shall follow the applications path(s).

#### 3-5.4.1.10 Connection Path

The Connection Path parameter shall contain one or more encoded paths as required by a combination of the service and the value of other parameters within the service data.

The Connection_Path shall be of the form:

| | |
|---|---|
| [[Electronic Key segment1]<br>[Network segment11]…[Network segment1N]<br>Port segment1] | *Port Segment Group for 1st router* |
| [[Electronic Key segment2]<br>[Network segment21]…[Network segment2N]<br>Port segment2] | *Port Segment Group for 2nd router* |
| … | |
| [[Electronic Key segmentM]<br>[Network segmentM1]…[Network segmentMN]<br>[[Application Path]<br>[Network segmentO1] … [Network segmentON]]<br>Port segmentM] | *Port Segment Group for Mth router* |
| [Electronic Key segmentP]<br>[Network segmentP1]…[Network segmentPN] | *Application Segment Group for target* |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Application Path1
[Application Path2]
[Application Path3]
[Simple Data segment]
[Network segmentQ1] … [Network segmentQN] [1]

NOTE: Network segments are allowed at the specified locations only if allowed by the specific network segment definition.

All segments listed in brackets […] are optional. See section C-1.4.2 for the definition of Electronic Key segment, section C-1.4.3 for the definition of Network Segment, section C-1.5 for the definition of the Application Path format and section C-1.4.5.1 for the definition of the Simple Data Segment.

The Connection Manager object of each intermediate and target node shall process all segments preceding Application Path(s) that are directed to it (see section 10-2). For the last-hop router (Mth router), the use of Network Segments that follow the Application Path must be directly associated with the object identified by that Application Path and that object must define its use.

Depending on the O→T_connection_parameters and T→O_connection_parameters fields and the presence of a data segment, one or more encoded application paths shall be specified. In general, the application paths are in the order of Configuration path, Consumption path, and Production path. However, a single encoded path can be used when configuration, consumption, and/or production use the same path. See Table 3-5.13 for the valid combinations and the implied meaning of the application paths. The application paths are relative to the *target* node.

[1] For the target, the Network Segment(s) that follow the Application Path(s) and the Data Segment are limited to certain Safety Network segments as defined in Volume 5 – CIP Safety.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-5.13 Encoded Application Path Ordering**

| Network Connection Parameters | | Data Segment Present | Number of Encoded Application Paths | | |
|---|---|---|---|---|---|
| **O→T Connection Type** | **T→O Connection Type** | | **1** | **2** | **3** |
| Null | Null | Yes | Path is for Configuration. | First path is for Configuration, second path is ignored | First path is for Configuration, second and third paths are ignored |
| | | No | Path is for pinging a device [1] | Invalid | Invalid |
| Not Null | Null | Yes | Path is for Configuration and Consumption. | First path is for Configuration, second path is for Consumption. | Invalid |
| | | No | Path is for Consumption. | Invalid | Invalid |
| Null | Not Null | Yes | Path is for Configuration and Production. | First path is for Configuration, second path is for Production. | Invalid |
| | | No | Path is for Production. | Invalid | Invalid |
| Not Null | Not Null | Yes | Path is for Configuration, Consumption and Production. | First path is for Configuration, second path is for Consumption and Production | First path is for Configuration, second path is for Consumption, third path is for Production |
| | | No | Path is for Consumption and Production. | First path is for Consumption, second path is for Production. | First path is ignored, second path is for Consumption, third path is for Production |

Table Footnotes:

1. If path is “20 01 24 01” used to ping a device, see section 3-5.4.2.2.1 Null Forward Open, Not Matching, Invalid otherwise.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 3-5.1 Device Diagram Showing Use of Connection Path Example Table 3-5.14 & Table 3-5.15**

**Table 3-5.14 Connection Path Example Showing Logical Segments, Received by 1st Router**

| Segment Groups | Segments | Description |
|---|---|---|
| 1st Port Segment Group | 34 04 | Key segment |
| | FF FF | Vendor |
| | 0C 00 | Device Type |
| | 21 00 | Product Code |
| | 81 01 | Major revision 1, Minor Revision 1, compatible keying |
| | 02 03 | Route out port 2 to link address 3 |
| Additional Port Segment groups | … | |
| Nth Port Segment group | 01 04 | Route out port 1to link address 4 (no key segment) |
| Application Segment group | 34 04 | Key Segment |
| | FF FF | Vendor |
| | 07 00 | Device Type |
| | 53 00 | Product Code |
| | 02 03 | Major Revision 2, Minor Revision 3, no compatible keying |
| | 43 14 | Production Inhibit Time Network Segment – 20ms |
| | 20 04 | Assembly object class |
| | 24 03 | Configuration Instance |
| | 2C 01 | O→T Connection point |
| | 2C 02 | T→O Connection point |
| | 80 01 | Data Segment (e.g.: 1 word of configuration data) |
| | 01 00 | Configuration Data |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-5.15 Connection Path Example Showing Logical Segments, Received by Target Device**

| Segment Groups | Segments | Description |
|---|---|---|
| Application Segment group | 34 04 | Key segment |
| | FF FF | Vendor |
| | 07 00 | Device Type |
| | 53 00 | Product Code |
| | 02 03 | Major revision 2, Minor Revision 3, no compatible keying |
| | 43 14 | Production Inhibit Time Network Segment – 20ms |
| | 20 04 | Assembly object class |
| | 24 03 | Configuration Instance |
| | 2C 01 | O→T Connection Point |
| | 2C 02 | T→O Connection Point |
| | 80 01 | Data Segment (e.g. 1 word of configuration data) |
| | 01 00 | Configuration Data |

This example includes 3 application paths, 20 04 24 03 is the configuration path, 20 04 2c 01 is the O→T path and 20 04 2c 02 is the T→O path (see Table 3-5.14).

**Figure 3-5.2 Device Diagram Showing Use of Connection Path Example in Table 3-5.16.**

**Table 3-5.16 Connection Path Example Showing a Symbolic Segment**

| Segment Groups | Segment | Description |
|---|---|---|
| Routing | 02 03 | Route out port 2 to MAC ID 03 |
| Information | 34 04 | Key segment |
| | FF FF | Vendor |
| | 0C 00 | Device Type |
| | 21 00 | Product Code |
| | 01 01 | Major revision 1, Minor Revision 1, exact match |
| | … | The routing information may continue for additional routers |
| | 01 04 | Route out port 1 (backplane) to slot 4 |
| Application Paths | 91 04 | 4 byte Symbol segment |
| | 74 61 | t a |
| | 67 31 | g 1 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This example includes one application path, a symbolic "tag1". At least one of the O→T and T→O network parameters connection types must not be NULL for this connection path to be valid (because there is no data segment present). If both the O→T and T→O network parameters connection types are not NULL then the both O→T and T→O application paths are symbolic "tag1" (see Table 3-5.16).

#### 3-5.4.1.10.1 Connection Path Compression

Refer to section C-1.6 for the description of Encoded Path Compression Rules. The following special case shortcut is defined, but only when the class is not the assembly object:

Shortcut connection path: Class A, Instance B, Connection Point C, Connection Point D, Data Segment.

For example: "20 aa 24 bb 2C cc 2C dd 80 …"

Full connection path: Class A, Instance B, Class A, Instance B, Connection Point C, Class A, Instance B, Connection Point D, Data Segment.

For example: "20 aa 24 bb 20 aa 24 bb 2C cc 20 aa 24 bb 2C dd 80 …"

#### 3-5.4.1.10.2 Assumed Assembly Object Attribute

When the Connection Path parameter finds a Configuration or I/O path consisting of an instance of the Assembly object (Class Code 4) without an attribute, the Data attribute (Attribute 3) is assumed.

An encoded path with the contents 20 04 24 xx 24 yy 24 zz followed by a Data Segment is decoded as follows:

- 20 04 24 xx 30 03 Configuration Path
- 20 04 24 yy 30 03 Consuming I/O Path
- 20 04 24 zz 30 03 Producing I/O Path

#### 3-5.4.1.11 Network Connection ID

The Network Connection ID shall be link specific and shall not be related to the connection serial number, which is connection specific and the same over all the links. The fields of the Network Connection ID shall be used to set the screening mechanism for the specified link. The Network Connection ID is either a CIP Produced Connection ID or CIP Consumed Connection ID.

A multicast CID shall not be reused until all connections associated with the CID have been closed or timed out.

#### 3-5.4.1.12 Transport Class and Trigger

The transport class and trigger specify the type of transport and the T→O production trigger required for the connection. See the TransportClass_trigger attribute of the Connection Object for the details of this parameter.

In addition, for Forward_Open services that establish a class 0/1 bound connection pair the following applies to the target and routers:

1. The transport class bits apply to both the O→T and T→O connections.
2. The direction bit should be Client (0) but in either case shall be ignored.
3. The trigger bits only apply to the T→O connection.

NOTE: Originator devices may or may not use the same trigger for the O→T connection.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-5.4.2 Forward_Open and Large_Forward_Open Request

The Forward_Open service (Service Code = $54_{hex}$) and Large_Forward_Open service (Service Code = $5B_{hex}$) are used to establish a Connection with a Target Device[1]. These services result in local connection establishment on each link along the path. The Large_Forward_Open service differs from the Forward_Open service only in the maximum size connection that can be established. This difference is evidenced in the data type and bit-field assignments of the O to T and T to O Network Connection parameters. All references to the Forward_Open service in this specification shall apply to both services.

A Forward Open can be either a non-Null Forward Open or a Null Forward Open. A non-Null Forward Open is a Forward_Open service request for which at least one of the Connection Types in the O→T or T→O network connection parameter field is not 00 (NULL). A Null Forward Open is a Forward_Open service request for which the Connection Type in both the O→T and T→O network connection parameter fields are both 00 (NULL) and results in no connection being established.

A Forward Open (both Null and non-Null) can be either not matching or matching. A matching Forward_Open service request received by the target device is one where the Connection Triad matches an existing connection.

The usage of each of the combinations of non-Null/Null and not matching/matching Forward Opens is described in the following sections and summarized in the list below.

1. Non-Null/not matching – open a connection
2. Non-Null/matching – error
3. Null/not matching
   - Ping a device
   - Configure
4. Null/matching – reconfigure

#### 3-5.4.2.1 Non-Null Forward Open

##### 3-5.4.2.1.1 Non-Null Forward Open, Not Matching

The Forward Open request sets up network, transport, and application connections. An application connection consists of a single transport connection, and one or two network connections that are in turn comprised of one or more multiple link connections. Each port segment in the connection path uses a link connection. The Forward_Open service between two devices builds one or two link connections as specified by the network connection parameter and the requested packet intervals (RPI). Since up to two network connections can be required for a single transport connection, they are differentiated by the O→T and T→O designations; O→T means originator to target, and T→O means target to originator.

A connection established to the Message Router object of the target (based on the Connection Path parameter in the Forward_Open request) results in an Explicit Messaging type connection. The format of the data sent on this connection uses the Message Router Request/Response Format as defined in Chapter 2-4 of this specification. Connections established to any other application object result in an I/O type connection. See the Connection Object attribute 2 (Instance Type).

[1] The processing of a single Forward_Open service may result in the creation of multiple active connection object instances.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The Forward_Open/Large_Forward_Open request parameters shown in Table 3-5.17 is the Request_Data sent for a Forward_Open/Large_Forward_Open service using the Message Router Request Format (see table 2-4.1). An originator shall not place an Electronic Key segment in the Message Router Request Request_Path for the Forward_Open/Large_Forward_Open services. Electronic key segments, if used, are included in the Connection_Path parameter of these services.

**Table 3-5.17 Forward_Open / Large_Forward_Open Request**

| Parameter Name | Data Type | Description |
|---|---|---|
| Priority/Time_tick | BYTE | Used to calculate request timeout information. See 3-5.4.1.3. |
| Time-out_ticks | USINT | Used to calculate request timeout information. See 3-5.4.1.3. |
| O→T Network Connection ID | UDINT | Network Connection ID to be used for the local link, originator to target. This is the originator's CIP Produced Connection ID. |
| T→O Network Connection ID | UDINT | Network Connection ID to be used for the local link, target to originator. This is the originator's CIP Consumed Connection ID. |
| Connection Serial Number | UINT | See **Object Specific Service Parameters** above. |
| Originator Vendor ID | UINT | Vendor ID of the originating node. |
| Originator Serial Number | UDINT | Serial Number of the originating node. |
| Connection Timeout Multiplier | USINT | See **Object Specific Service Parameters** above. |
| Reserved | octet | Reserved |
| | octet | |
| | octet | |
| O→T RPI | UDINT | Originator to Target requested packet rate, in microseconds. |
| O→T Network Connection Parameters | WORD/ DWORD [1] | See **Object Specific Service Parameters** above. |
| T→O RPI | UDINT | Target to Originator requested packet rate, in microseconds. |
| T→O Network Connection Parameters | WORD/ DWORD [1] | See **Object Specific Service Parameters** above. |
| Transport Class and Trigger | BYTE | See **Object Specific Service Parameters** above. |
| Connection_Path_Size | USINT | The number of 16 bit words in the *Connection_Path* field. |
| Connection_Path | Padded EPATH | Indicates the route to the Remote Target Device and/or its application. |

Table Footnotes:

1. The data type of this parameter is a WORD for the Forward_Open service, and a DWORD for the Large_Forward_Open service.

Success shall be returned when the connection requested has been established from this point forward in the path. This reply also shall indicate the connection serial number and the actual packet interval of the connection. Once the successful reply has been received, the connection shall be open from this point forward in the path. Targets shall wait at least 10 seconds after sending a successful response for the first packet on a connection.

#### 3-5.4.2.1.2 Non-Null Forward Open, Matching

A non-Null Forward Open for which the Connection Triad matches an existing connection shall return a general status = 0x01, extended status = 0x0100 (Connection in Use or Duplicate Forward Open).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The suggested originator behavior in the non-Null Forward Open, matching case should be to either close and re-establish the connection or wait for the connection to time-out and then establish the connection again. It shall be recognized that in the latter case, it shall take the connection 60 seconds (first data time-out) to time-out before the connection can be re-established.

#### 3-5.4.2.2 Null Forward Open

The Forward_Open service parameter values for Null Forward Opens are shown in Table 3-5.18.

**Table 3-5.18 Forward_Open/Large_Forward_Open Parameter Values for Null Forward Open**

| Parameter Name | Value |
|---|---|
| Priority/Time_tick | See section 3-5.4.1.3. |
| Time-out_ticks | See 3-5.4.1.3. |
| O→T Network Connection ID | Any valid value. |
| T→O Network Connection ID | Any valid value. |
| Connection Serial Number | See **Object Specific Service Parameters** above. |
| Originator Vendor ID | Vendor ID of the originating node. |
| Originator Serial Number | Serial number of originating node. |
| Connection Timeout Multiplier | Any valid value. |
| Reserved (3 octets) | Reserved |
| O→T RPI | Any valid value. |
| O→T Network Connection Parameters | Connection Size – Any valid value.<br>Fixed/Variable – Any valid value.<br>Priority – Any valid value.<br>Connection Type – 00 (**Null**).<br>Redundant Owner – Any valid value.<br>Reserved fields – 0. |
| T→O RPI | Any valid value. |
| T→O Network Connection Parameters | Connection Size – Any valid value.<br>Fixed/Variable – Any valid value.<br>Priority – Any valid value.<br>Connection Type – 00 (**Null**).<br>Redundant Owner – Any valid value.<br>Reserved fields – 0. |
| Transport Type/Trigger | Any valid value. |
| Connection_Path_Size | The number of 16 bit words in the *Connection_Path* field. |
| Connection_Path | Indicates the route to the Remote Target Device and/or its application. |

Uses of the Null Forward Open are documented in the sections below.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 3-5.4.2.2.1 Null Forward Open, Not Matching

A Null Forward Open for which the Connection Triad does not match an existing connection's parameters can be used for the following functions:

1 Ping a device, with the following characteristics:
   - The single application path "20 01 24 01" (i.e. Identity Object).
   - An electronic key segment may be included.
   - No data segment is included.
   - No connection is established.

2 Configure a device's application, with the following characteristics
   - A configuration application path and data segment shall be included in the request. The data is sent to the application specified by the path and applied.
   - If the entire configuration cannot be applied then none of the configuration shall be applied and the appropriate error code returned.
   - An electronic key segment may be included.
   - No connection is established.

A target device shall respond to a Null Forward Open request with one of the following errors when the target does not support the requested function ("ping a device" or "configure a device's application"):

- Null Forward Open Function Not Supported. General Status 0x01, Extended Status 0x0132 (recommended)
- Invalid Network Connection Parameter. General Status 0x01, Extended Status 0x0108 (deprecated)

#### 3-5.4.2.2.2 Null Forward Open, Matching

A Null Forward Open for which the Connection Triad matches an existing connection's parameters can be used to reconfigure a target device's application. Routers shall always forward these requests. The associated behaviors are:

- A configuration application path and data segment shall be included in the request and they are sent to the application to change the application configuration.
- If the entire configuration cannot be applied then none of the configuration shall be applied and the appropriate error code returned.
- The connection shall not be interrupted due to this request.

A target device shall respond to a Null Forward Open request with one of the following errors when the target does not support the requested function ("reconfigure a target device's application"):

- Null Forward Open Function Not Supported. General Status 0x01, Extended Status 0x0132 (recommended)
- Invalid Network Connection Parameter. General Status 0x01, Extended Status 0x0108 (deprecated)

NOTE: If the interpretation of the consume/produce data changes as a result of the reconfigure operation care must be taken in the producing and consuming applications. There is no specified mechanism to coordinate between the producing and consuming applications, so a change in the meaning of the real time data can result in unexpected operation. Devices have the option to reject a reconfiguration request, with an Object State Conflict error (General Status = 0x0C), to prevent this situation.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-5.4.3 Forward_Open and Large_Forward_Open Responses

The following format shall be used for all successful Forward_Open responses.

**Table 3-5.19 Successful Forward_Open / Large_Forward_Open Response**

| Parameter Name | Data Type | Description |
|---|---|---|
| O→T Network Connection ID | UDINT | Network Connection ID to be used for the local link, originator to target. If chosen by the originator, then the value is echoed back. This is the target's CIP Consumed Connection ID. |
| T→O Network Connection ID | UDINT | Network Connection ID to be used for the local link, target to originator. If chosen by the originator, then the value is echoed back. This is the target's CIP Produced Connection ID. |
| Connection Serial Number | UINT | Returns same value received in the Request packet. |
| Originator Vendor ID | UINT | Returns same value received in the Request packet. |
| Originator Serial Number | UDINT | Returns same value received in the Request packet. |
| O→T API | UDINT | Actual packet interval, originator to target, in microseconds. [1] |
| T→O API | UDINT | Actual packet interval, target to originator, in microseconds. [1] |
| Application Reply Size | USINT | Number of 16 bit words in the *Application Reply* field. |
| Reserved | USINT | Reserved |
| Application Reply | Array of Byte | Application specific data |

Table Footnotes:

1 If a Connection Object (0x05) is instantiated to support this connection:

- this value is used for the Connection Object Expected_packet_rate attribute, after converting to milliseconds.
- the Expected_packet_rate will not be used for the Connection Object Inactivity/Watchdog Timer. Refer to section 3-5.4.1.1.1 for connection timeout handling.

The following format shall be used for all Forward_Open failures. The requested connection shall not be established, and the object specific status words shall contain information about the reason for the failure. The remaining_path_size shall contain the length of the path at the point the connection request failed.

A duplicate Forward_Open service shall be defined as a Forward_Open service whose vendor_ID, connection_serial_number, and originator_serial_number match an existing connection's parameters. If the duplicate Forward_Open service is a null Forward_Open service (defined as the connection type in both the O→T and T→O network connection parameter fields are NULL), then the Forward_Open service shall be forwarded to the application for further processing. Null Forward_Open requests may be used to reconfigure the connection. The Connection Manager in the routers need not allocate additional resources for a duplicate Forward_Open request since the resources have already been allocated. If the duplicate Forward_Open request is not NULL, then a general status = 0x01, extended status = 0x0100 shall be returned.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-5.20 Unsuccessful Forward_Open / Large_Forward_Open Response**

| Parameter Name | Data Type | Description |
|---|---|---|
| Connection Serial Number | UINT | Returns same value received in the Request packet. |
| Originator Vendor ID | UINT | Returns same value received in the Request packet. |
| Originator Serial Number | UDINT | Returns same value received in the Request packet. |
| Remaining Path Size | STRUCT of: | This field is only present if the error was detected by a router. |
| | USINT | The number of words in the *Connection_Path* parameter of the request as received by the router that detects the error. |
| | octet | Reserved, shall be set to zero. |

Either a target device or an intermediate router shall return an Invalid Connection Size extended error code, along with the maximum connection size supported additional status word, if the connection size requested is not supported while processing a Large Forward_Open request.

In the failure response, the remaining remaining_path_size shall be the “pre-stripped” size. This shall be the size of the path when the node first receives the request and has not yet started processing it. A target node may return either the “pre-stripped” size or 0 for the remaining remaining_path_size.

### 3-5.4.4 Forward_Close

The Forward_Close Service (Service Code = $4E_{hex}$) is used to close a connection with a target device (and all other nodes in the connection path). A successful Forward_Close request and response shall result in all non-shared resources associated with this connection in all nodes participating in the original connection being released, including connection IDs, bandwidth allocation, and internal memory buffers. Shared resources are those still in use by other connection(s) due to multicast. A router should also release non-shared resources if the response received indicates an error of General Status 0x01 and Extended Status 0x0107 (Target Connection Not Found).

The Forward_Close request parameters shown in Table 3-5.21 is the Request_Data sent for a Forward_Close service using the Message Router Request Format (see table 2-4.1). An originator shall not place an Electronic Key segment in the Message Router Request Request_Path for the Forward_Close service. Electronic key segments, if used, are included in the Connection_Path parameter of this service.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-5.21 Forward_Close Service Request**

| Parameter Name | Data Type | Description |
|---|---|---|
| Priority/Time_tick | BYTE | Used to calculate request timeout information. |
| Time-out_ticks | USINT | Used to calculate request timeout information. |
| Connection Serial Number | UINT | Connection Serial Number of established connection. |
| Originator Vendor ID | UINT | Vendor ID of the originating node. |
| Originator Serial Number | UDINT | Serial Number of the originating node. |
| Connection_Path_Size | USINT | The number of 16 bit words in the *Connection_Path* field. |
| Reserved | USINT | Reserved |
| Connection_Path | Padded EPATH | Indicates the route to the Remote Target Device. |

For a Forward_Close service request received by a target node to be successful, it is sufficient that the Connection Triad matches an existing connection's parameters. If there is no connection match, no connection is released and the target shall return an error with a General Status code of 0x01 and an Extended Status code of 0x0107 (Target Connection Not Found), unless the device detected an improper Conection Path. The Connection Path Size parameters may be ignored by the target node. However, an improperly formatted Connection_Path or Connection_Path mismatch may cause the service to be unsuccessful and return an error regardless of a Connection Triad match (see connection path error conditions below).

A Forward_Close service request received by a router along the path between the originator and target nodes shall be forwarded regardless of a Connection Triad match (in the case of a match not found, the connection may have timed out at this router). An improperly formatted Connection_Path shall, and a Connection_Path mismatch may, cause the service to be unsuccessful and return an error (see connection path error conditions below).

If either a target or router does detect a connection path error condition, it shall return an error response as described below. In all cases the connection remains unchanged (the connection is not released).

Improperly Formatted Connection_Path:

- If a node detects that the value of the Connection_Path_Size field is smaller than the size of the Connection_Path, a General Status code of 0x15 shall be returned indicating that too much data is present in the service request.
- If a node detects that the value of the Connection_Path_Size field is greater than the size of the Connection_Path, General Status code 0x13 shall be returned indicating that not enough data is present in the service.

Connection_Path Mismatch:

- If the node detects a mismatch in the Connection_Path between the value received in the Forward_Close request and the value sent in the originating connection request, a General Status code of 0x01 and an Extended Status code of either 0x316 (Error In Forward Close Service Connection Path Mismatch) or 0x0315 (Invalid Segment in Connection Path) shall be returned.

Success shall be returned when the connection has been deleted at the target. The originator, and each router along the path, closes the connection and releases resources associated with that connection when the success response is received.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-5.22 Successful Forward_Close Response**

| Parameter Name | Data Type | Description |
|---|---|---|
| Connection Serial Number | UINT | Returns same value received in the Request packet. |
| Originator Vendor ID | UINT | Returns same value received in the Request packet. |
| Originator Serial Number | UDINT | Returns same value received in the Request packet. |
| Application Reply Size | USINT | Number of 16 bit words in the *Application Reply* field. |
| Reserved | USINT | Reserved |
| Application Reply | Array of byte | Application specific data |

**Table 3-5.23 Unsuccessful Forward_Close Response**

| Parameter Name | Data Type | Description |
|---|---|---|
| Connection Serial Number | UINT | Returns same value received in the Request packet. |
| Originator Vendor ID | UINT | Returns same value received in the Request packet. |
| Originator Serial Number | UDINT | Returns same value received in the Request packet. |
| Remaining Path Size | STRUCT of: | This field is only present if the error was detected by a router. |
| | USINT | The number of words in the *Connection_Path* parameter of the request as received by the router that detects the error. |
| | octet | Reserved, shall be set to zero. |

### 3-5.4.5 Unconnected_Send

The Unconnected_Send service shall allow an application to send a message to a device without first setting up a connection. The Unconnected_Send service shall use the Connection Manager object in each router to forward the message and to remember the return path. The UCMM of each link shall be used to forward the request from Connection Manager to Connection Manager just as it is for the Forward_Open service; however, no connection shall be built. The Unconnected_Send service shall be sent to the local Connection Manager and shall be sent between routers. When a router removes the last port segment, the embedded Message Request shall be formatted as a Message Router Request message (see Chapter 2-4) and sent to the port and link address of the last port segment using the UCMM for that link type.

NOTE: The target node never sees the Unconnected_Send service but only the embedded Message Request arriving via the UCMM.

#### 3-5.4.5.1 Unconnected_Send Request

The Unconnected_Send request parameters shown in Table 3-5.24 is the Request_Data sent for an Unconnected_Send service using the Message Router Request Format (see Table 2-4.1). A client shall not place an Electronic Key segment in the Message Router Request Request_Path for the Unconnected_Send service.

Electronic key segments, if used, are included in the Request_Path of the Embedded_Message_Request and Route_Path parameters of the Unconnected_Send service.

The parameters of an Unconnected_Send request are:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-5.24 Unconnected_Send Service Request Parameters**

| Parameter Name | | Data Type | Description |
|---|---|---|---|
| Priority/Time_tick | | BYTE | Used to calculate request timeout information. |
| Time-out_ticks | | USINT | Used to calculate request timeout information. |
| Embedded_Message_Request_Size | | UINT | Specifies the number of bytes in the Embedded Message Request. |
| Embedded_Message_Request | | Struct of | Embedded Message Router Request Format as defined in Chapter 2-4. |
| | Service | USINT | Service code of the request. |
| | Request_Path_Size | USINT | The number of 16 bit words in the Request_Path field (next element). |
| | Request_Path | Padded EPATH | This is an array of bytes whose contents convey the application path and other information of the request for this transaction. See Chapter 2-4.1.1. |
| | Request_Data | Array of octet | Service specific data to be delivered in the Explicit Messaging Request. If no additional data is to be sent with the Explicit Messaging Request, then this array will be empty. |
| Pad | | USINT | Only present if Embedded_Message_Request_Size is an odd value. |
| Route_Path_Size | | USINT | The number of 16 bit words in the Route_Path field. |
| Reserved | | USINT | Reserved byte. Shall be set to zero (0). |
| Route_Path | | Padded EPATH | Indicates the route to the Remote Target Device. See section 3-5.4.5.1.1. |

### 3-5.4.5.1.1 Route_Path

The route path shall be of the form:

| | |
|---|---|
| [Electronic Key segment1]<br>Port segment1 | Port Segment Group for 1st router |
| [[Electronic Key segment2]<br>Port segment2] | Port Segment Group for 2nd router |
| …<br>[[Electronic Key segmentN]<br>Port segmentN] | Port Segment Group for Nth router |

All segments listed in brackets […] are optional. If the Electronic Key segment is present the target must evaluate the key. See section C-1.4.1 for the definition of Port segment and C-1.4.2 for the definition of Electronic Key segment.

An Unconnected_Send example is shown in Table 3-5.25:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-5.25 Unconnected_Send Example using Figure 3-5.1 topology, Received by 1st Router**

| Parameters | | Values | Description |
|---|---|---|---|
| Service | | 52 | Unconnected_Send service code |
| Request_Path_Size | | 02 | Size of the request path |
| Request_Path | | 20 06 | Connection Manager class |
| | | 24 01 | Instance 1 |
| Priority/Time tick | | xx | Priority/Time tick, value not specified for this example |
| Time-out ticks | | xx | Time-out ticks, value not specified for this example |
| Embedded_Message_Request_Size | | 12 00 | The embedded message request size, in bytes, of the embedded message request |
| Embedded_Message_ Request | Service | 0E | Get_Attribute_Single service code |
| | Request_Path_ Size | 08 | The request path size, in 16 bit words |
| | Request_Path | 34 04 | Target device key segment |
| | | FF FF | Target Vendor Id |
| | | 07 00 | Target Device Type |
| | | 53 00 | Target Product Code |
| | | 02 03 | Target Major revision 2, Minor revision 3, exact match |
| | | 20 01 | Identity class |
| | | 24 01 | Instance 1 |
| | | 30 01 | Vendor ID, attribute 1 |
| | Request_Data | | No request data for a get attribute single |
| 1st Port Segment group | | 34 04 | 1st hop router key segment |
| | | FF FF | 1st hop router Vendor Id |
| | | 0C 00 | 1st hop router Device Type |
| | | 21 00 | 1st hop router Product Code |
| | | 81 01 | 1st hop router Major revision 1, Minor Revision 1, compatible keying |
| | | 02 03 | Route out port 2 of 1st hop router to link address 3 |
| Additional Port Segment groups | | … | |
| Nth Port Segment group | | 01 04 | NOTE: No key segment specified in this port segment group. Route out port 1 of Nth hop router to link address 4 |

### 3-5.4.5.2 Unconnected_Send Response

The Unconnected_Send response shall be generated by the last router from the UCMM response generated by the target node or by a router as the result of a UCMM time-out, a problem with the embedded message, or a problem with the Unconnected Service Request itself. The packet shall be routed from router to router using the information stored when the Unconnected_Send request was processed. The response shall contain a header with status information about the request and a variable length response generated by the target node.

The Reply Service code returned may be either the service code sent inside the Unconnected_Send service data *or* the Unconnected_Send service code. When a router detects the error, the Unconnected_Send Reply Service code 0xD2 shall be returned. When the target device detects the error, the embedded service code shall be returned (for example, if the requested service was a Get_Attribute_Single, then the Reply Service will be 0x8E). This is advantageous to originators and routers because:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- Of the two service codes (Unconnected Send and embedded) the service code that failed is returned to the originator.
- Routers are not required to parse the service code embedded within the Unconnected_Send request.

#### 3-5.4.5.2.1 Unconnected_Send Response, Successful

The Service Data associated with a successful Unconnected_Send response is:

**Table 3-5.26 Successful Unconnected_Send Response**

| Parameter Name | Data Type | Description |
|---|---|---|
| Service Response Data | Array of byte | This field contains the Explicit Messaging Service Data returned by the Target Device/Object. For example, this would contain Attribute Data in response to an embedded Get_Attribute_Single request. If the Explicit Messaging response returned by the Target Device/Object did not contain any Service Data, then this field shall be empty. |

#### 3-5.4.5.2.2 Unconnected_Send Response, Unsuccessful

The response Service Data associated with an unsuccessful Unconnected_Send response is defined below.

**Table 3-5.27 Unsuccessful Unconnected_Send Response**

| Parameter Name | Data Type | Description |
|---|---|---|
| Remaining Path Size | STRUCT of: | This field is only present if the error was detected by a router. |
| | USINT | The number of words in the *Route_Path* parameter of the request as received by the router that detects the error. |
| | octet | Reserved, shall be set to zero |

### 3-5.4.6 Get_Connection_Data

This service shall return the parameters associated with a specified connection_number. The connection_number may be different from device to device even for the same connection. The connection_number corresponds to the offset into the Connection Manager attribute that enumerates the status of the connections.

**Table 3-5.28 Get_Connection_Data Service Request**

| Parameter Name | Data Type | Description |
|---|---|---|
| Connection Number | UINT | Connection number |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-5.29 Get_Connection_Data Service Response**

| Parameter Name | Data Type | Description |
|---|---|---|
| Connection Number | UINT | |
| Connection State | UINT | |
| Originator Port | UINT | |
| Target Port | UINT | |
| Connection Serial Number | UINT | Connection Serial Number of established connection. |
| Originator Vendor ID | UINT | Vendor ID of the originating node. |
| Originator Serial Number | UDINT | Serial Number of the originating node. |
| Originator O→T CID | UDINT | |
| Target O→T CID | UDINT | |
| Connection Timeout Multiplier | USINT | |
| Reserved | USINT<br>USINT<br>USINT | |
| Originator RPI O→T | UDINT | |
| Originator API O→T | UDINT | |
| Originator T→O CID | UDINT | |
| Target T→O CID | UDINT | |
| Connection Timeout Multiplier | USINT | |
| Reserved | USINT<br>USINT<br>USINT | |
| Originator RPI T→O | UDINT | |
| Originator API T→O | UDINT | |

### 3-5.4.7 Get_Connection_Owner

The Get_Connection_Owner service of the Connection Manager shall return data about the connection that owns a particular object. It shall be implemented in any device that accepts redundant connections (see section 3-6.5.2). The format of the Get_Connection_Owner request shall be of the form:

**Table 3-5.30 Get_Connection_Owner Parameters**

| Parameter Name | Data Type | Description |
|---|---|---|
| Reserved | USINT | Shall be set to 0. |
| Connection_Path_Size | USINT | The number of 16 bit words in the Connection Path field. |
| Connection_Path | Padded EPATH | This shall be the same path as in a Forward_Open request. The target device shall remove the electronic key, network, and data segments before matching the path with the existing connections. |

The Get_Connection_Owner response shall be of the following form:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-5.31 Successful Get_Connection_Owner Response**

| Parameter Name | Data Type | Description |
|---|---|---|
| Number_of_Connections | USINT | The number of connections currently open to the specified path |
| Number_Claiming_Ownership | USINT | The number of connections currently claiming ownership. |
| Number_Ready_for_Ownership | USINT | The number of connections currently ready for ownership. |
| Last_Action | USINT | Last action values:<br>0 – there is no owning connection.<br>1 – the owning connection is in idle mode.<br>2 – the owning connection is in run mode.<br>3-254 – reserved.<br>255 – the implementation does not support one of these fields. |
| Connection_Serial_Number | UINT | The connection serial number from the Forward_Open request whose connection is currently the owner of the connection. If there is no owner the value shall be 0. |
| Originator_Vendor_ID | UINT | The Originator Vendor ID from the Forward_Open request whose connection is currently the owner of the connection. If there is no owner the value shall be 0. |
| Originator_Serial_Number | UDINT | The Originator Serial Number from the Forward_Open request whose connection is currently the owner of the connection. If there is no owner the value shall be 0. |

### 3-5.4.8 Search_Connection_Data

This service shall return the parameters associated with the specified connection serial number, originator vendor ID and originator serial number.

**Table 3-5.32 Search_Connection_Data Service Request**

| Parameter Name | Data Type | Description |
|---|---|---|
| Connection Serial Number | UINT | Connection Serial Number of established connection. |
| Originator Vendor ID | UINT | Vendor ID of the originating node. |
| Originator Serial Number | UDINT | Serial Number of the originating node. |

The format of the Search_Connection_Data response shall be the same as the response from the Get_Connection_Data service.

## 3-5.5 Connection Manager Object Instance Error Codes

The error codes are returned with the response to a Connection Manager Service Request that resulted in an error. These error codes shall be used to help diagnose the problem with a Service Request. The error code shall be split into an 8 bit general status and one or more 16-bit words of extended status. Unless specified otherwise, only the first word of extended status shall be required. Additional words of extended status may be used to specify additional device specific information. All devices that originate messages shall be able to handle multiple words of extended status.

The following table provides a summary of the available error codes. Only the type of node indicated in the Detected By column shall return the general status/extended status code. The Detected By column values are Originator, Router and/or Target. When detected by Router or Target devices these error codes will be contained in the response packet. When detected by Originator these error codes can be reported back to the application via an object interface (e.g. Connection Configuration object) or other internal interface.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-5.33 Connection Manager Service Error Codes**

| General Status | Extended Status | Detected By | Explanation and Description |
|---|---|---|---|
| 0x00 | | | Service completed successfully |
| 0x01 | 0x0000 through 0x00FF | | Obsolete. |
| 0x01 | 0x0100 | Router<br>Target | CONNECTION IN USE OR DUPLICATE FORWARD OPEN<br><br>This extended status code shall be returned when an originator is trying to make a connection to a target with which the originator may have already established a connection (Non-Null/matching Forward_Open – see section 3-5.4.2.1.2). |
| 0x01 | 0x0101 through 0x0102 | | Reserved by CIP |
| 0x01 | 0x0103 | Target | TRANSPORT CLASS AND TRIGGER COMBINATION NOT SUPPORTED<br>A transport class and trigger combination has been specified which is not supported by the target application. |
| 0x01 | 0x0104 | | Reserved by CIP |
| 0x01 | 0x0105 | See CIP Safety Specification (Volume 5 Chapter 3) | |
| 0x01 | 0x0106 | Target | OWNERSHIP CONFLICT<br>The connection cannot be established since another connection has exclusively allocated some of the resources required for this connection. An example of this would be that only one exclusive owner connection can control an output point on an I/O Module. If a second exclusive owner connection (or redundant owner connection) is attempted, this error shall be returned. |
| 0x01 | 0x0107 | Target | TARGET CONNECTION NOT FOUND<br>This extended status code shall be returned in response to the Forward_Close request, when the connection that is to be closed is not found at the target node. Routers shall not generate this extended status code. If the specified connection is not found at the router, the close request shall still be forwarded using the path specified in the Forward_Close request. |
| 0x01 | 0x0108 | Router<br>Target | INVALID NETWORK CONNECTION PARAMETER<br>This extended status code shall be returned as the result of specifying a connection type, connection priority, redundant owner or fixed / variable that is not supported by the device.<br>NOTE: This extended status code is deprecated. It is highly recommended that 0x011F, 0x0120, 0x0121, 0x0122, 0x0123, 0x0124, 0x0125 or 0x0132 be used instead. |
| 0x01 | 0x0109 | Router<br>Target | INVALID CONNECTION SIZE<br>This extended status code is returned when the target or router does not support the specified connection size. This could occur at a target because the size does not match the required size for a fixed size connection. It could occur at a router if the requested size is too large for the specified network.<br>An additional status word may follow indicating the maximum connection size supported by the responding node. The additional status word is required when issued in response to the Large_Forward_Open.<br>NOTE: This extended status code is deprecated. It is highly recommended that 0x0126, 0x0127 or 0x0128 be used instead. |
| 0x01 | 0x010A through 0x010F | | Reserved by CIP |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| General Status | Extended Status | Detected By | Explanation and Description |
|---|---|---|---|
| 0x01 | 0x0110 | Target | TARGET FOR CONNECTION NOT CONFIGURED<br>This extended status code shall be returned when a connection is requested to a target application that has not been configured and the connection request does not contain a data segment for configuration.<br>Reference section C-1.4.5 – DATA SEGMENT |
| 0x01 | 0x0111 | Router<br>Target | RPI NOT SUPPORTED.<br>This extended status code shall be returned if the device cannot support the requested O→T or T→O RPI. This extended status code may also be used if the connection time-out multiplier produces a time-out value that is not supported by the device or the production inhibit time is not valid.<br>NOTE: It is highly recommended to use Extended Status 0x0112 when the RPI values(s) are not acceptable.<br>NOTE: Use of this extended status code when the connection time-out multiplier is not supported is deprecated. It is highly recommended that 0x0133 be used instead.<br>NOTE: Use of this extended status code when the production inhibit time is not valid is deprecated. It is highly recommended that 0x011B be used instead. |
| 0x01 | 0x0112 | Router<br>Target | RPI VALUE(S) NOT ACCEPTABLE<br>This error is to be returned when the RPI value(s) in the forward open request are outside the range required by the application in the target device or the target is producing at a different interval. The target shall include information with acceptable RPI(s). For this error, the extended status size is 6 16-bit words and is formatted as follows: |

| Data Type | Value | Explanation of Field |
|---|---|---|
| UINT | 0x112 | Extended status code |
| USINT | variable | Acceptable Originator to Target RPI (see below) type, values:<br>0 – the RPI specified in the forward open was acceptable (the Originator to Target RPI value is ignored).[1]<br>1 – unspecified (used to suggest an alternate RPI, e.g. default)<br>2 – minimum acceptable RPI (used when RPI was too fast for range)<br>3 – maximum acceptable RPI (used when RPI was too slow for range)<br>4 – required RPI to correct mismatch (used when data already being consumed at a different interval)<br>5-255 – reserved |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| General Status | Extended Status | Detected By | Explanation and Description | | |
|---|---|---|---|---|---|
| | | | USINT | variable | Acceptable Target to Originator RPI (see below) type, values:<br>0 – the RPI specified in the forward open was acceptable (the Target to Originator RPI value is ignored).[1]<br>1 – unspecified (used to suggest an alternate RPI, e.g. default)<br>2 – minimum acceptable RPI (used when RPI was too fast for range)<br>3 – maximum acceptable RPI (used when RPI was too slow for range)<br>4 – required RPI to correct mismatch (used when data already being produced at a different interval, typically multicast)<br>5-255 – reserved |
| | | | UDINT | variable | Value of Originator to Target RPI that is within the acceptable range for the application. This field is defined the same as the RPI parameter in the forward open request. |
| | | | UDINT | variable | Value of Target to Originator RPI that is within the acceptable range for the application. This field is defined the same as the RPI parameter in the forward open request. |
| | | | 1 The value of the Originator to Target and Target to Originator type shall not both be 0. | | |
| 0x01 | 0x0113 | Originator<br>Router<br>Target | OUT OF CONNECTIONS<br>Connection Manager cannot support any more connections. The maximum number of connections supported by the Connection Manager has already been created. | | |
| 0x01 | 0x0114 | Router<br>Target | VENDOR ID OR PRODUCT CODE MISMATCH<br>The Product Code or Vendor ID specified in the electronic key logical segment does not match the Product Code or Vendor ID of the device. If the compatibility bit is set this error code is returned when the requested Vendor ID or Product Code is 0 or the device cannot emulate the specified Vendor ID or Product Code. | | |
| 0x01 | 0x0115 | Router<br>Target | DEVICE TYPE MISMATCH<br>The Device Type specified in the electronic key logical segment does not match the Device Type of the device. If the compatibility bit is set this error code is returned when the requested Device Type is 0 or the device cannot emulate the specified Device Type.<br>EXCEPTION: If the compatibility bit is set, Device Type 0 is allowed if the recipient can emulate a device that reports the deprecated Generic Device Type Number. | | |
| 0x01 | 0x0116 | Router<br>Target | REVISION MISMATCH<br>The major and minor revision specified in the electronic key logical segment does not correspond to a valid revision of the device. If the compatibility bit is set this error code is returned when the requested Major Revision and/or Minor Revision is 0 or the device cannot emulate the specified revision. | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| General Status | Extended Status | Detected By | Explanation and Description |
|---|---|---|---|
| 0x01 | 0x0117 | Target | INVALID PRODUCED OR CONSUMED APPLICATION PATH<br>The produced or consumed application path specified in the connection path does not correspond to a valid produced or consumed application path within the target application. This error could also be returned if a produced or consumed application path was required, but not provided by a connection request.<br>NOTE: This extended status code is deprecated. It is highly recommended that 0x012A, 0x012B or 0x012F be used instead. |
| 0x01 | 0x0118 | Target | INVALID OR INCONSISTENT CONFIGURATION APPLICATION PATH<br>An application path specified for the configuration data does not correspond to a configuration application or is inconsistent with the consumed or produced application paths. For example the connection path specifies float configuration data while the produced or consumed paths specify integer data.<br>NOTE: This extended status code is deprecated. It is highly recommended that 0x0129 or 0x012F be used instead. |
| 0x01 | 0x0119 | Target | NON-LISTEN ONLY CONNECTION NOT OPENED<br>Connection request fails since there are no non-Listen Only connections with the same T→O application path currently open. Refer to Section 3-6 for a description of application connection types<br>The extended status code shall be returned when an attempt is made to establish a Listen Only connection type to a target, which has no non-Listen Only connections with the same T→O application path already established. |
| 0x01 | 0x011A | Target | TARGET OBJECT OUT OF CONNECTIONS<br>The maximum number of connections supported by this instance of the target object has been exceeded.<br>For example, the Connection Manager could support 20 connections while the target object can only support 10 connections. On the 11th Connection Request to the target object, this extended status code would be used to signify that the maximum number of connections already exist to the target object. |
| 0x01 | 0x011B | Target | THE PRODUCTION INHIBIT TIME IS GREATER THAN THE RPI<br>The Production Inhibit Time is greater than the Target to Originator RPI. |
| 0x01 | 0x011C | Router<br>Target | TRANSPORT CLASS NOT SUPPORTED<br>The transport class requested in the Transport Type/Trigger parameter is not supported. |
| 0x01 | 0x011D | Router<br>Target | T→O PRODUCTION TRIGGER NOT SUPPORTED<br>The T→O production trigger requested in the Transport Type/Trigger parameter is not supported. |
| 0x01 | 0x011E | Target | DIRECTION NOT SUPPORTED<br>The direction requested in the Transport Type/Trigger parameter is not supported. |
| 0x01 | 0x011F | Target | INVALID ORIGINATOR TO TARGET NETWORK CONNECTION FIXVAR<br>This extended status code shall be returned as the result of specifying an O→T fixed / variable flag that is not supported. |
| 0x01 | 0x0120 | Target | INVALID TARGET TO ORIGINATOR NETWORK CONNECTION FIXVAR<br>This extended status code shall be returned as the result of specifying a T→O fixed / variable flag that is not supported. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| General Status | Extended Status | Detected By | Explanation and Description |
|---|---|---|---|
| 0x01 | 0x0121 | Target | INVALID ORIGINATOR TO TARGET NETWORK CONNECTION PRIORITY<br>This extended status code shall be returned as the result of specifying an O→T priority code that is not supported. |
| 0x01 | 0x0122 | Target | INVALID TARGET TO ORIGINATOR NETWORK CONNECTION PRIORITY<br>This extended status code shall be returned as the result of specifying a T→O priority code that is not supported. |
| 0x01 | 0x0123 | Target | INVALID ORIGINATOR TO TARGET NETWORK CONNECTION TYPE<br>This extended status code shall be returned as the result of specifying an O→T connection type that is not supported. |
| 0x01 | 0x0124 | Router<br>Target | INVALID TARGET TO ORIGINATOR NETWORK CONNECTION TYPE<br>This extended status code shall be returned as the result of specifying a T→O connection type that is not supported. |
| 0x01 | 0x0125 | Router<br>Target | INVALID ORIGINATOR TO TARGET NETWORK CONNECTION REDUNDANT_OWNER<br>This extended status code shall be returned as the result of specifying an O→T Redundant Owner flag that is not supported. |
| 0x01 | 0x0126 | Target | INVALID CONFIGURATION SIZE<br>This extended status code is returned when the target device determines that the data segment provided in the Connection_Path parameter did not contain an acceptable number of 16-bit words for the configuration application path requested.<br>An additional status word shall follow indicating the maximum configuration size supported.<br>**Data Type** / **Value** / **Explanation of Field**<br>UINT / 0x0126 / Extended Status Code<br>UINT / Size / Max Size in Words |
| 0x01 | 0x0127 | Router<br>Target | INVALID ORIGINATOR TO TARGET NETWORK CONNECTION SIZE<br>This extended status code is returned by the target when the size of the consuming object declared in the Forward_Open request and available on the target does not match the size declared in the O→T Network Connection Parameter.<br>This extended status code is returned by a router when it cannot support the size requested in the O→T Network Connection Parameter.<br>An additional status word shall follow indicating the maximum originator to target size supported.<br>**Data Type** / Value / **Explanation of Field**<br>UINT / 0x0127 / Extended Status Code<br>UINT / Size / Max Size in bytes |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| General Status | Extended Status | Detected By | Explanation and Description |
|---|---|---|---|
| 0x01 | 0x0128 | Router<br>Target | INVALID TARGET TO ORIGINATOR NETWORK CONNECTION SIZE<br>This extended status code is returned by the target when the size of the producing object declared in the Forward Open request and available on the target does not match the size declared in the T→O Network Connection Parameter.<br>This extended status code is returned by a router when it cannot support the size requested in the T→O Network Connection Parameter.<br>An additional status word shall follow indicating the maximum target to originator size supported. |
| | | | **Data Type** / **Value** / **Explanation of Field**<br>UINT / 0x0128 / Extended Status Code<br>UINT / Size / Max Size in bytes |
| 0x01 | 0x0129 | Target | INVALID CONFIGURATION APPLICATION PATH<br>The configuration application path specified in the connection path does not correspond to a valid configuration application path within the target application. This error could also be returned if a configuration application path was required, but not provided by a connection request. |
| 0x01 | 0x012A | Target | INVALID CONSUMING APPLICATION PATH<br>The consumed application path specified in the connection path does not correspond to a valid consumed application path within the target application. This error could also be returned if a consumed application path was required, but not provided by a connection request. |
| 0x01 | 0x012B | Target | INVALID PRODUCING APPLICATION PATH<br>The produced application path specified in the connection path does not correspond to a valid produced application path within the target application. This error could also be returned if a produced application path was required, but not provided by a connection request. |
| 0x01 | 0x012C | Target | CONFIGURATION SYMBOL DOES NOT EXIST<br>Configuration Symbol does not exist. The originator attempts to connect to a configuration tag name, but the name is not on the list of tags defined on the target. |
| 0x01 | 0x012D | Target | CONSUMING SYMBOL DOES NOT EXIST<br>Consuming Symbol does not exist. The originator attempts to connect to a consuming tag name, but the name is not on the list of tags defined on the target. |
| 0x01 | 0x012E | Target | PRODUCING SYMBOL DOES NOT EXIST<br>Producing Symbol does not exist. The originator attempts to connect to a producing tag name, but the name is not on the list of tags defined on the target. |
| 0x01 | 0x012F | Target | INCONSISTENT APPLICATION PATH COMBINATION<br>The combination of configuration and/or consume and/or produce application paths specified in the connection path are inconsistent with each other. |
| 0x01 | 0x0130 | Target | INCONSISTENT CONSUME DATA FORMAT<br>Information in the data segment is not consistent with the format of the consumed data. For example the configuration data specifies float configuration data while the consumed path specifies integer data. |
| 0x01 | 0x0131 | Target | INCONSISTENT PRODUCE DATA FORMAT<br>Information in the data segment is not consistent with the format of the produced data. For example the configuration data specifies float configuration data while the produced path specifies integer data. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| General Status | Extended Status | Detected By | Explanation and Description |
|---|---|---|---|
| 0x01 | 0x0132 | Target | NULL FORWARD OPEN FUNCTION NOT SUPPORTED<br>The target does not support the function requested by the Null Forward Open. The requested function may be “ping a device”, “configure a device’s application”, or “reconfigure a target device’s application”. |
| 0x01 | 0x0133 | Target<br>Router | CONNECTION TIMEOUT MULTIPLIER NOT ACCEPTABLE<br>This extended status code shall be returned as the result of specifying a connection timeout multiplier value that is reserved or that produces a timeout value that is too large to support in the device. |
| 0x01 | 0x0134 | Target | MISMATCHED T→O NETWORK CONNECTION SIZE<br>This extended status code shall be returned when an originator requests to open a Multicast connection to a T→O application path that is already being produced for another connection and the Size field in the T→O Network Connection Parameters in this request is valid but doesn’t match the existing connection’s Size field. |
| 0x01 | 0x0135 | Target | MISMATCHED T→O NETWORK CONNECTION FIXVAR<br>This extended status code shall be returned when an originator requests to open a Multicast connection to a T→O application path that is already being produced for another connection and the Fixed/Variable bit in the T→O Network Connection Parameters in this request is valid but doesn’t match the existing connection’s Fixed/Variable bit. |
| 0x01 | 0x0136 | Target | MISMATCHED T→O NETWORK CONNECTION PRIORITY<br>This extended status code shall be returned when an originator requests to open a Multicast connection to a T→O application path that is already being produced for another connection and the Priority field in the T→O Network Connection Parameters in this request is valid but doesn’t match the existing connection’s Priority field. |
| 0x01 | 0x0137 | Target | MISMATCHED TRANSPORT CLASS<br>This extended status code shall be returned when an originator requests to open a Multicast connection to a T→O application path that is already being produced for another connection and the Transport Class in this request is valid but doesn’t match the existing connection’s Transport Class. |
| 0x01 | 0x0138 | Target | MISMATCHED T→O PRODUCTION TRIGGER<br>This extended status code shall be returned when an originator requests to open a Multicast connection to a T→O application path that is already being produced for another connection and the T→O Production Trigger in this request is valid but doesn’t match the existing connection’s T→O Production Trigger. |
| 0x01 | 0x0139 | Target | MISMATCHED T→O PRODUCTION INHIBIT TIME SEGMENT<br>This extended status code shall be returned when an originator requests to open a Multicast connection to an application path that is already being produced for another connection and the PIT in this request is valid but doesn’t match the existing connection’s PIT. |
| 0x01 | 0x013A through 0x0202 | | Reserved by CIP |
| 0x01 | 0x0203 | Originator | CONNECTION TIMED OUT<br>This extended status code shall occur when a connection has timed-out. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| General Status | Extended Status | Detected By | Explanation and Description |
|---|---|---|---|
| 0x01 | 0x0204 | Originator<br>Router<br>Target | UNCONNECTED REQUEST TIMED OUT<br>An originator or router processing an Unconnected_Send, Forward_Open, Large_Forward_Open, or Forward_Close service shall return an Unconnected Request Timed Out error when the request cannot be delivered to the next device in the path (e.g the link specific retries fail), the allotted time is not sufficient to process the request, or no reply is received within the time specified by the Time_tick/Time-out_ticks parameters.<br>A target processing a Forward_Open, Large_Forward_Open, or Forward_Close service may return an Unconnected Request Timed Out error when the allotted time is not sufficient to process the request.<br>See section 3-5.4.1.3 Unconnected Request Timing for more information. |
| 0x01 | 0x0205 | Router | PARAMETER ERROR IN UNCONNECTED REQUEST SERVICE<br>For example, this shall be caused by a Connection Tick Time (see section 3-5.4.1.3) and Connection time-out combination in an Unconnected_Send, Forward_Open, or Forward_Close service that is not supported by a router. |
| 0x01 | 0x0206 | Originator<br>Router | MESSAGE TOO LARGE FOR UNCONNECTED_SEND SERVICE<br>This shall be caused when the Unconnected_Send is too large to be sent out on a network. |
| 0x01 | 0x0207 | Originator<br>Router | UNCONNECTED ACKNOWLEDGE WITHOUT REPLY<br>The message was sent via the unconnected message service and an acknowledge was received but a data response message was not received. |
| 0x01 | 0x0208 through 0x0300 | | Reserved by CIP |
| 0x01 | 0x0301 | Originator<br>Router<br>Target | NO BUFFER MEMORY AVAILABLE<br>The extended status code shall occur when insufficient connection buffer memory is available in the device. |
| 0x01 | 0x0302 | Originator<br>Router<br>Target | NETWORK BANDWIDTH NOT AVAILABLE FOR DATA<br>This extended status code shall be returned by any device in the path that is a producer and cannot allocate sufficient bandwidth for the connection on its link. This can only occur for connections that are specified as scheduled priority. |
| 0x01 | 0x0303 | Originator<br>Router<br>Target | NO CONSUMED CONNECTION ID FILTER AVAILABLE<br>Any device in the path that contains a link consumer for the connection and does not have an available consumed_connection_id filter available shall return this extended status code. |
| 0x01 | 0x0304 | Originator<br>Router<br>Target | NOT CONFIGURED TO SEND SCHEDULED PRIORITY DATA<br>If requested to make a connection that specifies scheduled priority, any device that is unable to send packets during the scheduled portion of the network update time interval shall return this extended status code. For example, on ControlNet this code shall be returned by a node whose MAC ID is greater than maximum scheduled node (SMAX). |
| 0x01 | 0x0305 | Router | SCHEDULE SIGNATURE MISMATCH<br>This extended status code shall be returned when the connection scheduling information in the originator device is not consistent with the connection scheduling information on the target network. |
| 0x01 | 0x0306 | Router | SCHEDULE SIGNATURE VALIDATION NOT POSSIBLE<br>This extended status code shall be returned when the connection scheduling information in the originator device cannot be validated on the target network. For example, on ControlNet this code shall be returned when there is no keeper in the master state. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| General Status | Extended Status | Detected By | Explanation and Description |
|---|---|---|---|
| 0x01 | 0x0307 through 0x0310 | | Reserved by CIP |
| 0x01 | 0x0311 | Originator<br>Router | PORT NOT AVAILABLE<br>A Port specified in a Port Segment is Not Available or does not exist. |
| 0x01 | 0x0312 | Originator<br>Router | LINK ADDRESS NOT VALID<br>Link Address specified in Port Segment Not Valid<br>This extended status code is the result of a port segment that specifies a link address that is not valid for the target network type. This extended status code shall not be used for link addresses that are valid for the target network type but do not respond. |
| 0x01 | 0x0313 through 0x0314 | | Reserved by CIP |
| 0x01 | 0x0315 | Originator<br>Router<br>Target | INVALID SEGMENT IN CONNECTION PATH<br>Invalid Segment Type or Segment Value in Connection Path<br>This extended status code is the result of a device being unable to decode the connection path. For example, this could be caused by an unrecognized path type or a segment type occurring unexpectedly. This extended status code shall only be used when no other more specific extended status code provided in this table applies. |
| 0x01 | 0x0316 | Router<br>Target | FORWARD CLOSE SERVICE CONNECTION PATH MISMATCH<br>The connection path in the Forward_Close service does not match the connection path in the connection being closed. This extended status error code has been deprecated because the Forward_Close service uses the connection triad for matching and doesn't use the connection path. |
| 0x01 | 0x0317 | Originator<br>Router<br>Target | SCHEDULING NOT SPECIFIED<br>Either the Schedule Network Segment was not present or the Encoded Value in the Schedule Network Segment is invalid (i.e. 0). |
| 0x01 | 0x0318 | Originator<br>Router | LINK ADDRESS TO SELF INVALID<br>Under some conditions (depends on the device), a link address in the Port Segment which points to the same device (loopback to yourself) is invalid. |
| 0x01 | 0x0319 | Router<br>Target | SECONDARY RESOURCES UNAVAILABLE<br>In a dual chassis redundant system, a connection request that is made to the primary system shall be duplicated on the secondary system. If the secondary system is unable to duplicate the connection request, then this extended status code shall be returned. |
| 0x01 | 0x031A | Target | RACK CONNECTION ALREADY ESTABLISHED<br>A request for a module connection has been refused because part of the corresponding data is already included in a rack connection. |
| 0x01 | 0x031B | Target | MODULE CONNECTION ALREADY ESTABLISHED<br>A request for a rack connection has been refused because part of the corresponding data is already included in a module connection. |
| 0x01 | 0x031C | Originator<br>Router<br>Target | MISCELLANEOUS<br>This extended status is returned when no other extended status code applies for a connection related error. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| General Status | Extended Status | Detected By | Explanation and Description |
|---|---|---|---|
| 0x01 | 0x031D | Target | REDUNDANT CONNECTION MISMATCH<br>This extended status code shall be returned when the following fields do not match when attempting to establish a redundant owner connection to the same target path:<br>- O→T_RPI;<br>- O→T_connection_parameters;<br>- T→O_RPI;<br>- T→O_connection_parameters;<br>- xport_type_and_trigger. |
| 0x01 | 0x031E | Target | NO MORE USER CONFIGURABLE LINK CONSUMER RESOURCES AVAILABLE IN THE PRODUCING MODULE<br>A target shall return this extended status when the configured number of consumers for a producing application are already in use. |
| 0x01 | 0x031F | Target | NO USER CONFIGURABLE LINK CONSUMER RESOURCES CONFIGURED IN THE PRODUCING MODULE<br>A target shall return this extended status when there are no consumers configured for a producing application to use. |
| 0x01 | 0x0320 – 0x07FF | | Vendor specific |
| 0x01 | 0x0800 | Originator<br>Router | NETWORK LINK OFFLINE<br>Network link in path to module is offline |
| 0x01 | 0x0801 – 0x080F | See CIP Safety Specification (Volume 5, Chapter 3) | |
| 0x01 | 0x0810 | Target | NO TARGET APPLICATION DATA AVAILABLE<br>This extended status code is returned when the target application does not have valid data to produce for the requested connection. |
| 0x01 | 0x0811 | Originator | NO ORIGINATOR APPLICATION DATA AVAILABLE<br>This extended status code is returned when the originator application does not have valid data to produce for the requested connection. |
| 0x01 | 0x0812 | Originator<br>Router | NODE ADDRESS HAS CHANGED SINCE THE NETWORK WAS SCHEDULED<br>A router on a scheduled network (e.g.: ControlNet) has a different node address than the value configured in the connection originator. |
| 0x01 | 0x0813 | Router<br>Target | NOT CONFIGURED FOR OFF-SUBNET MULTICAST<br>A multicast connection has been requested between a producer and a consumer that are on different subnets, and the producer is not configured for off-subnet multicast. |
| 0x01 | 0x0814 | Target | INVALID PRODUCE/CONSUME DATA FORMAT<br>Information in the data segment indicates that the format of the produced and/or consumed data is not valid.<br>NOTE: This extended status code is deprecated. It is highly recommended that 0x0130 or 0x0131 be used instead. |
| 0x01 | 0x0815 through 0xFCFF | | Reserved by CIP |
| 0x01 | 0xFD00 through 0xFFFF | | Do not use |
| 0x02 | Empty | Originator<br>Router | RESOURCE UNAVAILABLE FOR UNCONNECTED_SEND<br>The device lacks the resources to fully process the Unconnected Send Request. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| General Status | Extended Status | Detected By | Explanation and Description |
|---|---|---|---|
| 0x04 | Empty | Router | PATH SEGMENT ERROR IN UNCONNECTED SEND<br>Indicates the Router experienced a parsing error when extracting the Explicit Messaging Request from the Unconnected Send Request Service Data. |
| 0x09 | Index to Element | Target | ERROR IN DATA SEGMENT.<br>This general status code shall be returned when there is an error in the data segment of a forward open.<br>The Extended Status shall be the index to where the error was encountered in the Data Segment (Reference section C-1.4.5 – DATA SEGMENT). |
| 0x0C | Optional | Target | OBJECT STATE ERROR<br>This general status code shall be returned when the state of the target object of the connection prevents the service request from being handled. The Extended Status reports the object's present state. The extended status is optional.<br>For example, a target (application) object of the connection may need to be in an edit mode before attributes can be set. This is different from a service being rejected due to the state of the device. |
| 0x10 | Optional | Router<br>Target | DEVICE STATE ERROR<br>This general status code shall be returned when the state of the device prevents the service request from being handled. The Extended Status reports the device's present state. The extended status is optional.<br>For example, a controller may have a key switch which when set to the "hard run" state causes Service Requests to several different objects to fail (i.e. program edits). This general status code would then be returned. |
| 0x13 | None | Router<br>Target | NOT ENOUGH DATA<br>The service did not supply enough data to perform the specified operation. |
| 0x15 | None | Router<br>Target | TOO MUCH DATA<br>The service supplied more data than was expected. |

## 3-6 Application Connection Type using Class 0 or 1 Transports

The Forward Open service of the Connection Manager provides the ability to create two network connections at the same time, one Class 0/1 connection in the O→T direction and the other Class 0/1 connection in the T→O direction. When two connections are created in the same Forward Open service request certain behaviors are associated between the two connections depending on their application connection type.

The application type shall determine the target behavior concerning the relationship between different connections each sharing a producer, and shall be one of LISTEN_ONLY, INPUT_ONLY, EXCLUSIVE_OWNER, or REDUNDANT_OWNER.

The O→T application path, defined by the target, specifies the application type. In addition the Redundant Owner bit in the O→T Network Connection Parameters parameter must be used to differentiate between EXCLUSIVE_OWNER and REDUNDANT_OWNER. The device reflects the application type supported for each connection in the associated EDS ConnectionN entry in the EDS file and/or Target Connection List (0x4D) object instance. This enables a configuration tool to know the application type of each connection.

A further element in applying these connections is the real time formats, including Run/Idle notification.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-6.1 Real time formats including RUN/IDLE notification

Each class 0/1 network connection shall send packets using one of the following real time formats:

- Modeless format – no run/idle notification.
- Zero length data format indicates idle.
- Heartbeat format – no run/idle notification.
- 32-bit header format includes run/idle notification.
- Safety format (See Volume 5)

The real time format supported by a target device is indicated in the Connection Manager section, ConnectionN entry, Connection Parameters field of its EDS file. This allows a connection originator configuration tool to know the O→T and T→O real time formats for each connection.

The connection originator needs to know the real time formats to produce data using the correct format and to correctly decode the data it is consuming. The O→T and T→O real time formats are specified in instance attribute 2 of the Connection Configuration Object.

The target device needs to know the real time format to decode the data it is consuming and to produce data using the correct format. The target device knows the real time formats based on the Connection_Path specified in the Forward_Open, each unique Connection_Path is mapped to a O→T real time format and a T→O real time format.

#### 3-6.1.1 Modeless Format

The modeless real time format may include 0-n bytes of application data and there is no run/idle notification included with this real time format. The modeless real time format network connection may be either fixed or variable size.

A class 0 modeless real time packet format is:

| |
|---|
| 0-n bytes of application data |

A class 1 modeless real time packet format is:

| | |
|---|---|
| 2 byte sequence count | 0-n bytes of application data |

#### 3-6.1.2 Zero Length Data Format

When the zero length data real time format includes 1-n bytes of application data the producer is indicating run mode. When the zero length data real time format includes 0 bytes of application data the producer is indicating idle mode. The zero length data format network connection shall be variable size.

A class 0 zero length data real time packet format showing idle mode is:

| |
|---|
| 0 bytes of application data |

A class 0 zero length data real time packet format showing run mode is:

| |
|---|
| 1-n bytes of application data |

A class 1 zero length data real time packet format showing idle mode is:

| | |
|---|---|
| 2 byte sequence count | 0 bytes of application data |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

A class 1 zero length data real time packet format showing run mode is:

| | |
|---|---|
| 2 byte sequence count | 1-n bytes of application data |

### 3-6.1.3 Heartbeat Format

The heartbeat real time format includes 0 bytes of application data and there is no run/idle notification included with this real time format. The heartbeat real time format network connection shall be fixed size.

A class 0 heartbeat real time packet format is:

| |
|---|
| 0 bytes of application data |

A class 1 heartbeat real time packet format is:

| | |
|---|---|
| 2 byte sequence count | 0 bytes of application data |

### 3-6.1.4 32-Bit Header Format

The 32 bit header real time format includes 0-n bytes of application data prefixed with 32 bits of header. The 32-bit header real time format network connection may be either fixed or variable size.

The 32-bit real time header format prefixed to the real-time data shall be the following form:

**Table 3-6.1 32-Bit Real Time Header**

| Bits 4-31 | Bits 2-3 | Bit 1 | Bit 0 |
|---|---|---|---|
| Reserved | ROO | COO | Run/Idle |

The run/idle flag (bit 0) shall be set (1 = RUN) to indicate that the following data shall be sent to the target application. It shall be clear (0 = IDLE) to indicate that the idle event shall be sent to the target application. The ROO and COO fields (bits 1-3) are defined in Section 3-6.5.2, Redundant Owner. The reserved field (bits 4-31) shall be reserved and set to 0.

A class 0 32-bit header real time packet format is:

| | |
|---|---|
| 32-bit real time header | 0-n bytes of application data |

A class 1 32-bit header real time packet format is:

| | | |
|---|---|---|
| 2 byte sequence count | 32-bit real time header | 0-n bytes of application data |

## 3-6.2 Configuration

Any application type connection may send configuration with a Forward_Open. If present the configuration shall be specified in a data segment and shall be delivered to the component specified in the configuration path. If no data segment is specified the existing configuration continues to be used. If a data segment is specified and no other active I/O connection which specified this component exists, the configuration in the data segment shall be applied. If a data segment is specified and another I/O connection to this component exists the configuration in the data segment shall be compared to the existing configuration; if there are no differences the connection may be opened, if there are differences an Ownership Conflict error (0x01,0x106) shall be returned.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-6.3 T→O Network Connection Rules

There are T→O behaviors based on the Connection Type and Application Type. There are two basic cases:

- Creating a new connection
- Joining an existing connection/transport

Table 3-6.2 describes the cases when the connection request will be opened or rejected based on the T→O Connection Type and Application Type, assuming that all other Forward_Open parameters are correct and supported, including segments in the Connection path. If a connection is supported on more than one port, Table 3-6.3 specifies the rules when a request is received on a port that does not already have a T→O transport for the connection.

A device is not required to support both Point to Point and Multicast connections. However if the device supports both Point to Point and Multicast connections, then it shall support both connection types to the same Application Path at the same time as long as there are resources to open the new connection.

If the target does not have enough resources for the new connection, then the appropriate error from Table 3-5.33 Connection Manager Service Error Codes shall be returned. Examples of error codes include, but are not limited to, General Status 0x01 with an Extended Status of 0x0113, 0x011A, 0x0301, 0x0302, 0x031E or 0x031F.

**Table 3-6.2 Connection Request Establishment T→O Behavior – Same Port**

| | | New T→O Connection Request | | | |
|---|---|---|---|---|---|
| | | **Point to Point Listen Only** | **Multicast Listen Only** | **Point to Point non-Listen Only** | **Multicast non-Listen-Only** |
| **Existing T→O Network Connection** | **Point to Point connection** | Succeeds, opens a new point to point T→O connection [1] | Succeeds, opens a new multicast T→O connection [1] | Succeeds, opens a new point to point T→O connection | Succeeds, opens a new multicast T→O connection |
| | **Multicast connection** | Succeeds, opens a new point to point T→O connection [1] | Succeeds, reuses the existing multicast T→O transport [1] | Succeeds, opens a new point to point T→O connection | Succeeds, reuses the existing multicast T→O transport |
| | **No existing connection** | Fails, returns general status 0x01, extended status 0x0119 | Fails, returns general status 0x01, extended status 0x0119 | Succeeds, opens a new point to point T→O connection | Succeeds, opens a new multicast T→O connection |

Table Footnotes:

1. The new Listen Only connection is dependent on an existing non-Listen Only connection.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-6.3 Connection Request Establishment T→O Behavior – Different Port**

| | | New T→O Connection Request On Different Port | | | |
|---|---|---|---|---|---|
| | | **Point to Point Listen Only** | **Multicast Listen Only** | **Point to Point non-Listen Only** | **Multicast non-Listen Only** |
| **Existing T→O Network Connection** | **Point to Point or Multicast connection** | Succeeds, opens a new point to point T→O connection[1] on new port | Succeeds, opens a new multicast T→O connection[1] on new port | Succeeds, opens a new point to point T→O connection on new port | Succeeds, opens a new multicast T→O connection on new port |

Table Footnotes:

1. The new Listen Only connection is dependent on an existing non-Listen Only connection.

### 3-6.3.1 T→O Multicast Matching Rules

For multicast if an established T→O producer exists on the same port, the T→O parameters listed in the Table 3-6.4 below shall match. If not, the device shall return the indicated error. See Table 3-5.33 Connection Manager Service Error Codes for detailed status code information.

**Table 3-6.4 T→O Parameter Matching Error Codes**

| T→O parameter | General Status | Extended Status |
|---|---|---|
| T→O RPI | 0x01 | 0x0112 |
| T→O Network Connection Parameters – Size | 0x01 | 0x0134 |
| T→O Network Connection Parameters – Fixed/Variable | 0x01 | 0x0135 |
| T→O Network Connection Parameters – Priority | 0x01 | 0x0136 |
| Transport Class | 0x01 | 0x0137 |
| T→O Production Trigger | 0x01 | 0x0138 |
| T→O Production Inhibit Time | 0x01 | 0x0139 |

## 3-6.4 Specifying Different Application Connection Types

The EDS section (chapter 7) of this specification defines how to declare the application connection type for a connection. Software tools can use this information to audit and enforce connection type issues if desired (issues like multiple exclusive owner connections to the same target, only listen only connections to a target, exclusive owner connection with redundant owner connection to the same target, etc.).

There is no requirement for connection originator devices or bridge/router devices to take action based on connection type (other than redundant owner connections), so the connection type is not explicitly included in the forward open data (with the exception of redundant owner connections). A specific connection path shall have a single application connection type. The target device can then take the appropriate actions for the connection based on the application connection type definitions below.

## 3-6.5 Application types

The requirements for each of the 4 connection types, Listen Only, Input Only, Exclusive Owner and Redundant Owner are detailed in this section.

The number of simultaneous connections is implementation dependent. When the target cannot accept more connections it shall respond with appropriate general/extended status.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-6.5.1 Listen Only

A Listen Only connection type is one that may provide application data in the T→O direction and a heartbeat in the O→T direction. If a connection has an application type of Listen Only, it shall be dependent on a non-Listen Only application connection for its existence. If the target receives a request to open a Listen Only connection when there is no existing non-Listen Only connection for the requested T→O application Path, the target shall respond with Non-Listen Only Connection Not Opened, General Status = 0x01, Extended Status = 0x0119.

The O→T connection shall use the heartbeat format as described in section 3-6.1.3.

If the device also supports an Input Only connection to the same T→O application path then the Listen Only and Input Only O→T heartbeat application paths shall be different.

Devices that wish to listen to multicast data without providing configuration may use this application type.

If the last connection on which a Listen Only connection depends is closed or times out, the target device shall stop sending the T→O data for the Listen Only connection resulting in a time out at the originator device.

### 3-6.5.2 Input Only

An Input Only connection type is one that may provide application data in the T→O direction and a heartbeat in the O→T direction. If a connection has an application type of Input Only, it shall not be dependent on any other connection for its existence.

The O→T connection shall use the heartbeat format as described in section 3-6.1.3.

If the device also supports a Listen Only connection to the same T→O application path then the Listen Only and Input Only O→T heartbeat application paths shall be different.

### 3-6.5.3 Exclusive Owner

An Exclusive Owner connection type is one that may provide application data in both the T→O direction and the O→T direction. If a connection has an application type of Exclusive Owner, it shall not be dependent on any other connection for its existence.

O→T application data that controls outputs may be present. A target shall only accept one Exclusive Owner connection which specifies the same O→T application path.

The term connection owner shall refer to the connection originator whose O→T packets are being consumed by the target object. The term owning connection shall refer to the connection associated with the connection owner.

When an Exclusive Owner connection timeout occurs in a target device, the target device shall stop sending the associated T→O data. The T→O data must not be sent even if one or more Input Only connections exist. This requirement is necessary to signal the originator of the Exclusive Owner connection that the O→T data is no longer being received by the target device.

NOTE: One possible way to prevent an Exclusive Owner connection timeout from stopping the T→O production for other network connections to the same T→O application path, is for the target device to support production of the T→O data as point to point for the Exclusive Owner connection and simultaneously support multicast production for other network connections to the same T→O application path.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-6.5.4 Redundant Owner

#### 3-6.5.4.1 General

A Redundant Owner connection type is one that may provide application data in both the T→O direction and the O→T direction. The Redundant Owner connection shall allow multiple separate originator applications to each establish an independent, identical connection to the transport of a target application. The target transport shall in turn send events to the target application so that the Redundant Owner connection appears as a single, Exclusive Owner connection to the target application. At most one of the originator applications shall have its application data applied to the target application. The target transport shall multicast the target application data to each of the originator applications.

When a single Redundant Owner connection exists and a Redundant Owner connection timeout occurs in a target device, the target device shall stop sending the associated T→O data. The T→O data must not be sent even if one or more Input Only connections exist. This requirement is necessary to signal the originator of the Redundant Owner connection that the O→T data is no longer being received by the target device.

NOTE: One possible way to prevent a Redundant Owner connection timeout from stopping the T→O production for other network connections to the same T→O application path, is for the target device to support production of the T→O data as point to point for the redundant owner connection and simultaneously support multicast production for other network connections to the same T→O application path.

NOTE: The Redundant Owner connection may commonly be used in applications where up time is at a premium. In these applications, multiple originators can each establish a connection to a target (typically an output application). Should an originator fail or otherwise give up control, another originator can quickly have its application data applied to the target application without having to establish a connection with the target.

#### 3-6.5.4.2 Establishing the Connection

A redundant owner connection and an exclusive owner connection shall not both be established to a target application at the same time. Multiple redundant owner connections may be established to the same target simultaneously provided that the following fields of the Forward_Open request are identical and the connection paths match. The connection path shall match including any data segments.

- O→T_RPI;
- O→T_connection_parameters;
- T→O_RPI;
- T→O_connection_parameters;
- xport_type_and_trigger.

The target shall return general status = 0x01 and extended status = 0x031D if any of these fields do not match.

The target transport shall only send the CM_open_indication to the target application when the first Forward_Open is received. Subsequent redundant Forward_Opens shall not cause a CM_open_indication to be sent to the target application.

NOTE: Bit 15 of the Forward Open request.O→T_connection_parameters specifies whether the connection is an exclusive owner or redundant owner connection


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 3-6.5.4.3 Redundant Owner O→T Data Format

#### 3-6.5.4.3.1 General

The Redundant owner shall use the 32-bit header format described in section 3-6.1.4.

#### 3-6.5.4.3.2 Claim Output Ownership (COO) Flag

The COO flag shall be set (1) when an originator application wants its connection to be the owning connection of the target application. The COO flag shall be reset (0) when an originator application does not want its connection to be the owning connection of the target application. When the owning connection resets (0) its COO flag, its sibling connections shall be checked for a set (1) COO flag. The new owner shall be any of the connections that have their COO flag set.

NOTE: This results in undefined behavior if more than one other connection has its COO flag set.

#### 3-6.5.4.3.3 Ready for Ownership of Outputs (ROO) Priority Value

The ROO priority value shall be non-zero when an originator application does not want to force its connection to be the owning connection of the target application, but is ready to be the owning connection should there be no originator applications claiming to be the owning connection. The ROO priority value shall be zero when the originator application does not want to be the owning connection of the target connection and is not to be the owning connection should there be no originator application claiming to be the owning connection. The ROO priority value shall be used only when the COO flag is reset.

The value of the ROO field can range from 0 to 3. The originator applications shall each determine a unique non-zero ROO value.

### 3-6.5.4.4 Determining the owning connection

The originator applications shall determine among themselves which originator application has the owning connection. The owning connection shall be determined by the originator application that sets its COO flag. In situations where multiple originator applications have their COO flag set or where no originator connections have their COO flag set, the following rules shall be applied by the target transport to determine the owning connection.

- There shall be no owning connection until an originator application sends a real-time packet with the COO flag set;
- If there is only one originator application which had the COO flag set in its last realtime packet, that originator application shall have the owning connection;
- If there are multiple originator applications which had the COO flag set in its last realtime packet, the last originator application that transitioned its COO flag from reset to set shall have the owning connection.
- If the originator application with the owning connection resets its COO flag, closes its connection, or if that connection times out, and no other originator applications have their COO flags set, the originator application with the highest non-zero ROO priority value shall have the owning connection.
- If all of the originator applications have their COO flags reset and ROO priority values set to zero, there shall be no owning connection.
- When the first real-time packet containing a set COO flag is received by the target transport, the originator application that sent the real-time packet shall have the owning connection.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 3-6.5.4.5 Transporting events and data to a target application

The connection related events from each of the redundant owner connections shall be combined using the following rules such that the target application sees only a single exclusive owner connection:

- Until an owning connection is initially determined, the transport shall not indicate realtime data reception to the target application;
- If an owning connection is determined, the transport shall indicate the event consistent with the owning connections real-time data to the target application. If the Run/Idle flag is reset in the real-time data for the owning connection, the transport shall indicate the idle state to the target application. If the Run/Idle flag is set in the real-time data for the owning connection, the transport shall indicate the run state and the real-time data to the target application;
- If an owning connection had been previously determined, but no originator is currently claiming or ready for ownership, the transport shall indicate idle state to the target application;
- If all the redundant connections are closed or have been timed out, the target transport shall indicate the event consistent with the last connection to have been closed or timed out to the target application.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 3-7 Port Object Class Definition

**Class Code: F4 hex**

The Port Object describes the communication interfaces that are present on the device and visible to CIP. A tool/originator can use the Port Object to learn the types of subnets and/or backplanes which can be reached via CIP routing (see Chapter 10) through a device as well as which object provides access to its link-specific functionality. In addition a tool/originator can use the Port Object to learn the CIP routing capabilities (see Port Object Instance attribute 10).

In many cases, a single Port instance is used to directly reflect a physical port (e.g. DeviceNet). In some cases, multiple Port instances share a single physical port (e.g. an EtherNet/IP Port and Modbus/TCP Port may share the same Ethernet interface). And there are cases where there are multiple physical ports for a single Port instance (e.g. a dual-port EtherNet/IP device with DLR).

The Port Object provides the port name, port number within the device, and other port information for each communication interface on the device.

Implementation requirements:

1. One instance shall exist for each CIP routable port.
2. If the device supports multiple ports of the same Port Type (see instance attribute 1), one instance shall exist for each of them.
3. The Port Object is required if another object definition requires it (for example the Originator Connection List Object).
4. Instances may exist for non-routable ports.
5. Instances may exist for devices that support a single port. NOTE: A single port device may support routing by allowing an incoming packet to be routed back out the port, in this case one instance shall exist for the port.

NOTE: Devices that do not support a CIP routable port and do not support multiple ports of the same Port Type are not required to support the Port Object (unless rule 3 applies).

**Table 3-7.1 Revision History**

| Port Class Revision | Description |
|---|---|
| 01 | Initial Release |
| 02 | Instance Attribute 10 added – Port Routing Capabilities |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 3-7.1 Port Object Class Attributes

The Port Object Class attributes are defined below in Table 3-7.2.

**Table 3-7.2 Port Class Attributes**

| Attr ID | Need In Implementation | Access Rule | Attribute Name | Data Type | Attribute Description | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Get | Revision | UINT | Revision of this object | The current value assigned to this attribute is two (2). |
| 2 | Required | Get | Max Instance | UINT | Maximum instance number. | |
| 3 | Required | Get | Num Instances | UINT | Number of ports currently instantiated. | |
| 4 – 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |
| 8 | Required | Get | Entry Port | UINT | Returns the instance of the Port Object that describes the port through which this request entered the device. | |
| 9 | Required | Get | Port Instance Info | ARRAY of STRUCT of | Array of structures containing instance attributes 1 and 2 from each instance. | The array is indexed by instance number starting with zero, up to the maximum instance number. The values for instance zero and any non-instantiated instances shall be zero. |
| | | | Port Type | UINT | Enumerates the type of port | See instance attribute #1 |
| | | | Port Number | UINT | CIP port number associated with this port | See instance attribute #2 |

## 3-7.2 Port Object Instance Attributes

The following list provides a summary of the Port Instance attributes and their associated data types.

**Table 3-7.3 Port Object Instance Attributes**

| Attr ID | Need In Implem | Access Rule | NV | Attribute Name | Data Type | Description of Attribute | Semantics |
|---|---|---|---|---|---|---|---|
| 1 | Required | Get | NV | Port Type | UINT | Enumerates the type of port | See section 3-7.2.1.1 |
| 2 | Required | Get | NV | Port Number | UINT | CIP port number associated with this port | See section 3-7.2.1.2 |
| 3 | Required | Get | NV | Link Object | STRUCT of | | See section 3-7.2.1.1 |
| | | | | Path Length | UINT | Number of 16 bit words in the following path[1] | |
| | | | | Link Path | Padded EPATH | Logical path segments that identify the object for this port. | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need In Implem | Access Rule | NV | Attribute Name | Data Type | Description of Attribute | Semantics |
|---|---|---|---|---|---|---|---|
| 4 | Required | Get | NV | Port Name | SHORT_ STRING | String which names the communications interface. The maximum number of characters in the string is 64. For example, this may be "Port A". | See section 3-7.2.1.3 |
| 5 | Optional | Get | NV | Port Type Name | SHORT_ STRING | String which names the port type. The maximum number of characters in the string is 64. | See section 3-7.2.1.1 |
| 6 | Optional | Set | NV | Port Description | SHORT_ STRING | String which describes the port. The maximum number of characters in the string is 64. For example, this may be "Product Line 22". | See section 3-7.2.1.4 |
| 7 | Required | Get | NV | Port Number and Node Address | Padded EPATH | This is a single Port Segment containing the Port Number of this port and the Link Address of this device on this port. | See section 3-7.2.1.5 |
| 8 | Conditional[2] | Get | NV | Port Node Range | STRUCT of | | See section 3-7.2.1.6 |
| | | | | Minimum Node Number | UINT | Minimum node number on port. | |
| | | | | Maximum Node Number | UINT | Maximum node number on port. | |
| 9 | Conditonal[3] | Get | NV | Chassis Identity | Packed EPATH | Electronic key of chassis this port is attached to.<br>This attribute shall be a single Logical Electronic Key Segment with Format 4 of the Logical Electronic Key segment. The Vendor ID, Device Type, Product Code, Major Revision and Minor Revision fields shall not be 0. The Compatibility field shall be 0 (indicating match). | See section 3-7.2.1.7 |
| 10 | Required | Get | NV | Port Routing Capabilities | DWORD | Bit string that defines the routing capabilities of this port | See section 3-7.2.1.8 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need In Implem | Access Rule | NV | Attribute Name | Data Type | Description of Attribute | Semantics |
|---|---|---|---|---|---|---|---|
| 11 | Conditional [4] | Get | NV | Associated Communication Objects | STRUCT of | List of communication object instances associated with this Port Object instance | See section 3-7.2.1.9 |
| | | | | | USINT | Number of entries in array | |
| | | | | | Array of STRUCT of | | |
| | | | | | USINT | Number of 16 bit words in the following path | |
| | | | | | Padded EPATH | Logical path segments that identify an associated communication object instance | |

Table Footnotes:

1. The Path Length may be 0 if there is no link object associated with this port instance. If a link object exists for the port instance, the path shall be specified.
2. If a device can report its port characteristics within the range allowed (e.g. DeviceNet MACID) then it shall support this attribute. Otherwise (e.g. EtherNet/IP IP Address) it shall not support this attribute.
3. If this port is attached to a chassis then this attribute is optional, otherwise it is not allowed.
4. If there is more than one Port Object instance of this Port Type (attribute 1), this attribute is required, otherwise this attribute is optional.

## 3-7.2.1 Semantics

### 3-7.2.1.1 Port Type, Link Object and Port Type Name, Attributes 1, 3 and 5

The vendor assigns values to these three attributes to indicate the type of the port, whether or not it supports routing, and whether it provides a link specific object to make link specific functionality visible to CIP.

When a port provides routing the corresponding Port Type (1- 65534), Link Object and Port Type Name values assigned to the supported technology in Table 3-7.4 shall be used.

When a port does not provide routing the Port Type value shall be set to 0 and the corresponding Link Object and Port Type Name values assigned to the supported technology in Table 3-7.4 shall be used.

When a port cannot report its type information based on its state (e.g. the user must choose which protocol it should support), the Port Type value shall be set to 65535. See Port Type 65535 in Table 3-7.4 for the corresponding Link Object and Port Type Name values.

The Port Type values and their associated Link Object classes and Port Type Name values are listed in Table 3-7.4.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 3-7.4 Port Type and Associated Link Object Classes and Port Type Name Values**

| Port Technology | Port Type (attribute 1) | Link Object (attribute 3) | | Port Type Name (attribute 5) |
|---|---|---|---|---|
| | | Link Object Name | Class Code | |
| Any technology below (1 through 65534), but routing is not supported | 0 | Link object assigned to underlying technology or null. | Class code of assigned link object (if implemented) | Name assigned to underlying technology (1-65534) below. |
| Reserved for legacy use[1] | 1 | Vendor Specific or null | Vendor Specific (if implemented) | Vendor assigned |
| ControlNet as defined in Volume 4 | 2 | ControlNet Object | 0xF0 | ControlNet |
| ControlNet with media redundancy as defined in Volume 4 | 3 | ControlNet Object | 0xF0 | ControlNet Redundant |
| EtherNet/IP as defined in Volume 2 | 4 | TCP/IP Interface Object | 0xF5 | EtherNet/IP [2] |
| DeviceNet as defined in Volume 3 | 5 | DeviceNet Object | 0x03 | DeviceNet |
| Reserved for legacy use[1] | 6-99 | Vendor Specific or null | Vendor Specific (if implemented) | Vendor Assigned |
| Vendor Specific Types 100-199 | 100-199 | Vendor Specific or null | Vendor Specific (if used) | Vendor Assigned |
| CompoNet as defined in Volume 6 | 200 | CompoNet Link Object | 0xF7 | CompoNet |
| Modbus/TCP as defined in Volume 7 | 201 | TCP/IP Interface Object or null | 0xF5 | Modbus/TCP |
| Modbus/SL as defined in Volume 7 | 202 | Vendor specific or null. | Vendor Specific (if implemented) | Modbus/SL |
| SERCOS III as defined in Volume 5 | 203 | Sercos III Link Object | 0x4C | SERCOS III |
| Reserved for future use | 204-65534 | Reserved for future use | | |
| Any technology above (1 through 65534), but the type of port is user configurable | 65535 | Link object assigned to underlying technology or null. | Class code of assigned link object (if implemented) | Unconfigured port |

Table Footnotes:

1. Port types marked as "Reserved for legacy use" indicate port types that were defined prior to the publication of this specification. They are not defined in this specification. Devices should not implement these port types without prior knowledge of the legacy usage.
2. This port type was formerly known as TCP/IP

The link object path value (attribute 3) shall consist of one logical class segment and one logical instance segment. The maximum size is 12 bytes. See Appendix C, Logical Segments.

#### 3-7.2.1.2 Port Number, Attribute 2

Manufacturer assigns a unique value to identify each communication port. Value 0 is reserved and cannot be used.

Value 1 is reserved for a backplane port. Devices with no backplane port or a backplane that does not support CIP communications shall not have a Port Number 1. The Port Number for the backplane port of a device that supports a single backplane port shall be 1. The Port Number for one of the backplane ports of a device that supports multiple backplane ports shall be 1 and the others can be any other valid port number.

The Port Number value is encoded in the Port Segment defined in Appendix C-1.4.1, Port Segment, and is used for routing.

#### 3-7.2.1.3 Port Name, Attribute 4

This attribute is the vendor assigned name of the communications interface associated with this instance.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This value is unique for each communications interface. If multiple CIP ports use the same communications interface, they shall each have the same Port Name value. Likewise, a CIP port shall not have the same Port Name value as any other CIP port if they do not share the same communications interface.

When there is a one-to-one relationship between a communications interface and a physical network port and the port has a user visible label on the product, the value should be equivalent to the label (e.g. use the textual name for a symbol).

When there is a one-to-many relationship between an internal communication interface and multiple physical network ports and each port has a user visible label on the product, this attribute should report the internal interface name (typically, the link object instance associated with each physical port reports the user visible label names in this case).

#### 3-7.2.1.4 Port Description, Attribute 6

The default value of the Port Description attribute shall be an empty string. This attribute is used by a client to add a description to their port.

#### 3-7.2.1.5 Port Number and Node Address, Attribute 7

The Port Number and Node Address value shall be a Port Segment containing the Port Number of this port and the Link Address of this device on the port. The encoded port number shall match the value presented in attribute 2.

For devices that do not have a Link Address on this port (for example, an EtherNet/IP port with no assigned IP address) or for Port Type 0 (no CIP routing), the Link Address portion of the Port Segment shall be zero length. For example, for Port Number 2, a zero-length Link Address would be indicated by 0x12 0x00.

See Appendix C-1.4.1, Port Segment for details on the Port Segment including examples.

#### 3-7.2.1.6 Port Node Range, Attribute 8

This attribute indicates the lowest and highest link address on the Port Type (attribute 1) that this port is connected to. This attribute can be used by a browsing tool to limit the range of link addresses browsed. For instance, for a 17 slot backplane, this attribute would have the values 0 and 16, for DeviceNet this attribute would have the values 0 and 63.

#### 3-7.2.1.7 Chassis Identity, Attribute 9

This attribute may be used to indicate the identity of the chassis that this port is currently plugged into. This attribute is useful when browsing. For example, the browser can identify the chassis that the device is plugged into, may load the chassis EDS file and associated icon, and use this identification to determine what additional modules can be plugged into this chassis. If this attribute is not implemented, then browsing clients will not have a standardized way of identifying the chassis, and will have to represent the chassis generically or use vendor-specific methods.

This attribute should be used to indicate the Identity of the chassis as accurately as possible. However, due to technical limitations of some chassis implementations, it may not be possible or practical to accurately determine certain attributes of the chassis, including version, slot ranges, and specific part numbers. In this case, the vendor may choose to include the identity of a generic chassis or perhaps the largest available chassis.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 3-7.2.1.8 Port Routing Capabilities, Attribute 10

This attribute indicates the specific routing capabilities of this port.

The Port Routing Capabilities attribute shall be all 0's if no routing is supported.

See Table 3-7.5 for the definition of the routing capability bits. If the bit is set, the port supports the specified routing function. In this table, 'incoming' refers to routing requests received on this port to any port on the device, and 'outgoing' refers to routing requests received from any port out this port.

**Table 3-7.5 Port Routing Capabilities Attribute Bit Definitions**

| Bit | Routing Capability |
|---|---|
| 0 | Routing of incoming Unconnected Messaging supported |
| 1 | Routing of outgoing Unconnected Messaging supported |
| 2 | Routing of incoming Transport Class 0/1 Connections supported |
| 3 | Routing of outgoing Transport Class 0/1 Connections supported |
| 4 | Routing of incoming Transport Class 2/3 Connections supported |
| 5 | Routing of outgoing Transport Class 2/3 Connections supported |
| 6 | Routing of outgoing DeviceNet CIP Safety Connections supported[1] |
| 7 – 31 | Reserved |

Table Footnotes:

1. Valid only for DeviceNet ports. See Volume 5, CIP Safety, for details.

#### 3-7.2.1.9 Associated Communication Objects, Attribute 11

The list should include any open and/or vendor specific objects that impact the operation of the communications path associated with this instance of the object. Each entry in the Array of Padded EPATH shall specify a single class segment followed by a single instance segment. If this attribute is supported, every Communication Object associated with this port instance shall be included. The following list is an example of open objects that are classified as a Communication Object. Table 5A-1.1 defines if an open object is a Communications object

1. DeviceNet Object – 03 hex
2. Modbus Object – 44 hex
3. Modbus Serial Link Object – 46 hex
4. Device Level Ring Object – 47 hex
5. QoS Object – 48 hex
6. SERCOS III Link Object – 4C hex
7. RSTP Bridge Object – 54 hex
8. RSTP Port Object – 55 hex
9. Parallel Redundancy Protocol Object – 56 hex
10. PRP Nodes Table Object – 57 hex
11. ControlNet Object – F0 hex
12. ControlNet Keeper Object – F1 hex
13. ControlNet Scheduling Object – F2 hex
14. TCP/IP Interface Object – F5 hex
15. Ethernet Link Object – F6 hex
16. CompoNet Link Object – F7 hex
17. CompoNet Repeater Object – F8 hex


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Associated Communication Objects shall be instances other than 1 for all Port Object instances having more than one instance of the same Port Type. This prevents old clients that may have assumed instance one for a communication object from retrieving incorrect information.

## 3-7.3 Common Services

The Port Object provides the following Common services:

**Table 3-7.6 Port Object Common Services**

| Service Code | Need In Implementation | | Service Name | Service Description |
|---|---|---|---|---|
| | Class | Instance | | |
| 01hex | Optional | Optional | Get_Attributes_All | Returns the contents of all attributes |
| 0Ehex | Required | Required | Get_Attribute_Single | Returns the contents of the specified attribute |
| 10hex | n/a | Conditional [1] | Set_Attribute_Single | Modifies an attribute |

Table Footnotes:

1. Required if any settable attributes are implemented

### 3-7.3.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 3-7.7** Get_Attributes_All Response Data – Class Level

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 [1] |
| 2 | UINT | Max Instance | |
| 3 | UINT | Number of Instances | |
| 8 | UINT | Entry Port | |
| 9 | ARRAY of:<br>STRUCT of: | Port Instance Info | |
| | UINT | Port Type | |
| | UINT | Port Number | |

Table Footnotes:

1. The default value only applies for Revision 1 of this object. Newer revisions require that this attribute be implemented, which will override this default value.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 3-7.8** Get_Attributes_All Response Data – Instance Level

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Port Type | |
| 2 | UINT | Port Number | |
| 3 | STRUCT of: | Link Object | |
| | UINT | Path Length | |
| | Padded EPATH | Link Path | |
| 4 | SHORT_STRING | Port Name | |
| 7 | Padded EPATH | Port Number and Node Address | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Volume 1: Common Industrial Protocol (CIP™) Specification

# Chapter 4: CIP Object Model


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Contents


4-1 Introduction ..... 3
4-2 Description ..... 4
4-3 Class Code ..... 4
4-4 Attributes ..... 4
4-4.1 Class Attributes ..... 4
4-4.2 Instance Attributes ..... 6
4-5 Common Services ..... 7
4-5.1 Get_Attributes_All Response ..... 8
4-5.2 Set_Attributes_All Request ..... 8
4-5.3 Optional Services ..... 9
4-6 Object–specific Services ..... 9
4-6.1 Service Parameters ..... 11
4-6.2 Service Response Data ..... 11
4-7 Behavior ..... 11
4-7.1 State ..... 12
4-7.2 Event ..... 13
4-8 Accessing Application object data ..... 13
4-8.1 Access Through Explicit Messaging Connections ..... 14
4-8.2 Access Through I/O Connections ..... 14
4-9 Extending Existing Objects and Defining New Objects ..... 16
4-9.1 CIP Object Address Ranges ..... 17
4-9.2 Making Extensions to Open Objects ..... 18
4-9.2.1 Vendor–specific Extensions ..... 18
4-9.2.1.1 Example ..... 19
4-9.2.1.2 Implementing a Vendor–specific Extension ..... 20
4-9.2.2 Open Extensions ..... 21
4-9.3 Defining a New Object ..... 21
4-9.3.1 Vendor–specific ..... 22
4-9.3.2 Open ..... 23
4-9.4 New Common Service ..... 23



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 4-1 Introduction

This chapter includes an explanation of how to read the object specifications in the library. Each object is defined using the same format.

| For information about | Go to Section |
|---|---|
| Reading an Object Specification | 4-1 |
| Description of Object | 4-2 |
| Class Code | 4-3 |
| Attributes | 4-4 |
| Common Services | 4-5 |
| Object–specific Services | 4-6 |
| Behavior | 4-7 |
| Connections | 4-8 |

Each specification contained in the library is defined based on the contents of an object.

An object consists of the following. See Figure 4-1.1.

- a set of closely related attributes (data)
- a set of behaviors
- a set of services (common or object–specific)
- a set of connections

**Figure 4-1.1 Defining an Object Specification**

The objects in the CIP Application Object Library are defined in the following terms:

**Table 4-1.1 Object Definitions**

| | |
|---|---|
| **Description** | a description of the object being specified |
| **Class Code** | a hexadecimal identifier assigned to each CIP object |
| **Attributes** | the data associated with this object |
| **Common Services** | a list of the common services defined for this object |
| **Object–specific Services** | the full specifications of any services unique to this object |
| **Connections** | connections supported by this object |
| **Behavior** | the relationship between attribute values and services |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 4-2 Description

Every object specification begins with a brief functional definition of the object being defined. For example, the Description of the Identity Object reads: *This object provides identification of and general information about the device.*

## 4-3 Class Code

This part of the definition is a hexadecimal value unique to an object. Use the class code to identify the object class when accessing objects in devices. ODVA is responsible for allocation and coordination of the class codes. However, managing Vendor Specific class codes and guaranteeing the values are unique to a Vendor ID is the responsibility of individual member companies.

In addition to the individual object definition, a summary list of all the objects and their class codes is located at the beginning of the Object Library.

## 4-4 Attributes

The Attribute part of an object specification is divided into two sections:

- Class attributes
- Instance attributes

### 4-4.1 Class Attributes

A *Class Attribute* is an attribute whose scope is that of the class as a whole, rather than any one particular instance. Therefore, the list of Class Attributes is different than the list of Instance Attributes. CIP defines the Instance ID value zero (0) to designate the Class level versus a specific Instance within the Class. Class Attributes are defined using the following terms:

**Table 4-4.1 Class Attribute Fields**

| Attribute ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

1. **Attribute ID** is an integer identification value assigned to an attribute. Use the Attribute ID in the Get_Attributes and Set_Attributes services list. The Attribute ID identifies the particular attribute being accessed.

2. **Need in Implementation** specifies whether or not the attribute is necessary in the object class implementation. An attribute may be O*ptional, Required or Conditional.*

   A conditional attribute is required if certain object behaviors and/or attributes are implemented as defined by the class or within the Device Profile.

   If an attribute is *optional*, then you must define a default value in the semantics.

   An optional attribute's default value shall provide the same behavior as if the attribute was not implemented.

   If the default value of an optional attribute does not match the behavior of the object when the object does not implement the attribute, the object definition shall declare the behavior.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

In all cases, the term "default" indicates a "factory default" as shipped from the vendor.

3. The **Access Rule** specifies how a requestor can access an attribute. The definitions for access rules are:
   - Settable (Set) – The attribute can be accessed by at least one of the set services. For enumerated attributes, if the device supports only one of the values and the object definition does not require more than one value to be supported, then the attribute may be implemented as Get only.

     **Important:** Settable attributes, unless otherwise specified by the object definition, shall also be accessed by get services.
   - Gettable (Get) – The attribute shall be accessed by at least one of the get services.
4. **NV** indicates whether an attribute value is maintained through power cycles. This column is used in object definitions where non-volatile storage of attribute values is required. An entry of '**NV**' indicates value shall be saved, '**V**' means not saved.
5. **Name** refers to the attribute.
6. **Data Type** - See Appendix C-2.
7. **Description of Attribute** provides general information about the attribute.
8. **Semantics of Values** specifies the meaning of the value of the attribute.

**Important:** Seven Class Attribute IDs are reserved for class object definitions. They are:

- Revision
- Max Instance
- Number of Instances
- Optional Attribute list
- Optional Service list
- Maximum Number Class Attributes
- Maximum Number Instance Attributes

Because these attributes are reserved, attribute ID numbers 1 through 7 are **always** reserved. Therefore, if you want to add a class attribute to an object definition, you must start with attribute ID #8. Unless otherwise specified in an object definition, each class attribute numbered 1 through 7 exists as an optional attribute within each class.

The seven reserved Class Attributes have the following definitions:

**Table 4-4.2 Reserved Class Attributes for All Object Class Definitions**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Conditional * | Get | Revision | UINT | Revision of this object | The current value assigned to this attribute is one (01). If updates that require an increase in this value are made, then the value of this attribute increases by 1. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 2 | Optional | Get | Max Instance | UINT | Maximum instance number of an object currently created in this class level of the device. | The largest instance number of a created object at this class hierarchy level. |
| 3 | Optional | Get | Number of Instances | UINT | Number of object instances currently created at this class level of the device. | The number of object instances at this class hierarchy level. |
| 4 | Optional | Get | Optional attribute list | STRUCT of | List of optional instance attributes utilized in an object class implementation. | A list of attribute numbers specifying the optional attributes implemented in the device for this class. |
| | | | number of attributes | UINT | Number of attributes in the optional attribute list. | The number of attribute numbers in the list. |
| | | | optional attributes | ARRAY of UINT | List of optional attribute numbers. | The optional attribute numbers. |
| 5 | Optional | Get | Optional service list | STRUCT of | List of optional services utilized in an object class implementation. | A list of service codes specifying the optional services implemented in the device for this class. |
| | | | number services | UINT | Number of services in the optional service list. | The number of service codes in the list. |
| | | | optional services | ARRAY of UINT | List of optional service codes. | The optional service codes. |
| 6 | Optional | Get | Maximum ID Number Class Attributes | UINT | The attribute ID number of the last class attribute of the class definition implemented in the device. | |
| 7 | Optional | Get | Maximum ID Number Instance Attributes | UINT | The attribute ID number of the last instance attribute of the class definition implemented in the device. | |

* If the value is 01, then this attribute is OPTIONAL in implementation. If the value is greater than 01, then this attribute is REQUIRED.

### 4-4.2 Instance Attributes

An *Instance Attribute* is an attribute whose value is unique to an object instance and whose definition is shared by all instances of an object. Each instance need only support the optional attributes that apply to it. If an instance does not support an optional attribute, the Attribute Not Supported (General Status code 0x14) error shall be returned for services targeting that attribute.

Instance Attributes are defined in the same terms as Class Attributes. There are no reserved Instance Attributes.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 4-4.3 Instance Attribute Fields**

| Attribute ID | Need in Implementation | Access Rule | NV | Name | CIP Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

## 4-5 Common Services

Common Services are those whose request/response parameters and required behaviors are defined in Appendix A.

The Common Service component of an object definition includes:

**Table 4-5.1 Common Service Fields**

| Service | Need in Implementation | | Service | Description of |
|---|---|---|---|---|
| Code | Class | Instance | Name | Service |
| 1 | 2 | 3 | 4 | 5 |

1. **Service Code** is the hexadecimal value assigned to each CIP service.

2-3. **Need in implementation** specifies whether or not the service is needed in the implementation of this object at the **Class** level or at the **Instance** level. In these columns will appear one of four specifications:

- Optional; or
- Required; or
- Conditional; or
- Not applicable (“n/a”)

A conditional service is required if certain object behaviors, attributes and/or services are implemented as defined by the object. All conditional services shall specify the condition(s) that qualify the service to be required. If the need in implementation is specified as Not Applicable at the class or instance level, it shall not be supported at that level.

**Important:** If an optional service is implemented in a class, and the optional Service List class attribute is also implemented in the class, the service shall be included in the Service List.

Services trigger the Behavior of an object based on the values of the attributes accessed by the service. Common Services can be directed to either the Class level or the Instance level of an object, which may produce different behavior at each level.

- *Class Level:* behavior triggered by services sent to the Object Class (Instance ID zero).
- *Instance Level:* behavior triggered by services sent to the Object *Instance*.

**Table 4-5.2 Common Service Naming Conventions**

| Common Services sent to | Are called |
|---|---|
| the Class Level of an object | Common Class Level Services |
| the Instance Level of an object | Common Instance Level Services |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

4. **Service Name** refers to the service. See Appendix A for a complete list of Common Services.
5. **Description of Service** provides a brief definition of the service.

### 4-5.1 Get_Attributes_All Response

When the Get_Attributes_All common service is included in the list of supported common services, then the Get_Attributes_All response shall be included in the object definition. This specifies the structure of the data returned in the Service Data portion of Explicit Response Message. (This is the Response_Data field described in the Message Router Response Format. See Chapter 2 for the definition.) The following list is an example of how the structure of the Service Data portion of a Get_Attributes_All response is typically specified:

**Table 4-5.3 Example Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 |
| 2 | UINT | Max Instance | 0 |
| 3 | UINT | Number of Instances | 0 |
| 4 | STRUCT of: | Optional attribute List | |
| | UINT | number of attributes | 0 |
| | ARRAY of UINT | optional attributes | (null) |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

The following rules are to be adhered to when specifying an object's Get_Attributes_All response format:

- The Get_Attributes_All response definition shall supply default values for all Optional and Conditional attributes. The use of default values is described in section A-4.1 Get_Attributes_All.
- If new attributes are added to the response structure, those attributes shall be added to the end.

For CIP defined (public) objects, the definition of the Get_Attributes_All response data (e.g: see Table 5A-2.10, Get_Attributes_All Response – Class Level for the Identity object) is controlled by CIP. Inclusion of vendor specific or any other data in the response data is not allowed.

### 4-5.2 Set_Attributes_All Request

When the Set_Attributes_All common service is included in the list of supported common services, then the Set_Attributes_All request must be included in the object definition. This specifies the sequence or order of the data supplied in the Service Data portion of the request. (This is the Request_Data field described in the Message Router Request Format. See Chapter 2 for the definition.) The following list is an example of how the format of the Service Data portion of a Set_Attributes_All request is typically specified:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 4-5.4 Example Set_Attributes_All Request Data**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Output Range | |
| 4 | USINT | Value Data Type | |
| 10 | UINT | Fault State | 0 |
| 11 | UINT | Idle State | 0 |
| 15 | UINT | Fault Value | 0 |
| 32 | UINT | Idle Value | 0 |

**Important:** The device shall accept the Default Value for all unsupported (optional) attributes.

The following rules are to be adhered to when specifying an object's Set_Attributes_All request format:

- Default Values shall be specified for all "optional" attributes that are not implemented. If a product chooses not to implement an optional attribute, it shall accept the specified Default Value for the unimplemented attribute.
- If new attributes are added to the request structure, those attributes shall be added to the end.

For CIP defined objects, the definition of the Set_Attributes_All service data is controlled by CIP. Inclusion of vendor specific or any other data in the service data is not allowed.

### 4-5.3 Optional Services

All objects shall have the following services defined as Optional at both the Class and Instance level unless otherwise specified by an object class as Conditional or Required. Services not listed here or in the object definition shall not be supported.

- Get_Attribute_Single
- Set_Attribute_Single
- Get_Attribute_List
- Set_Attribute_List
- Multiple_Service_Packet
- No_Operation (NOP)
- Get_Member
- Set_Member
- Insert_Member
- Remove_Member

## 4-6 Object–specific Services

The section titled "Object–specific Services" provides a list of the unique services supported by this class of objects and their service codes.

Whereas Common Services can be used in many objects, Object–specific Services are unique to each object.

For example, many objects support the Get_Attributes_All service; however, only the *DeviceNet Object* supports Allocate_Master/Slave_Connection_Set (see DeviceNet Specification, Volume 3).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The Object–specific Services component of the object class definition includes:

**Table 4-6.1 Object-specific Service Data Fields**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 1 | 2 | 3 | 4 | 5 |

1. **Service Code** is the hexadecimal value assigned to each CIP service.

2-3. **Need in implementation** specifies whether or not the service is needed in the implementation of this object at the **Class** level or at the **Instance** level. In these columns will appear one of four specifications:

- Optional; or
- Required; or
- Conditional; or
- Not applicable (“n/a”)

A conditional service is required if certain object behaviors, attributes and/or services are implemented as defined by the object. All conditional services shall specify the condition(s) that qualify the service to be required. If the need in implementation is specified as Not applicable at the class or instance level, it shall not be supported at that level.

**Important:** If a service is specified to be optional and is implemented in a device, its service code must be included in the “optional service list” class attribute.

Services trigger the Behavior of an object based on the values of the attributes accessed by the service. Object–specific Services can be directed to either the Class level or the Instance level of an object, which may produce different behavior at each level.

- Class Level: behavior triggered by services sent to the Object Class.
- Instance Level***:*** behavior triggered by services sent to the Object *Instance*.

**Table 4-6.2 Object-specific Service Naming Convention**

| Object–specific Services sent to | Are called |
|---|---|
| the Class Level of an object | Object–specific Class Level Services |
| the Instance Level of an object | Object–specific Instance Level Services |

4. **Service Name** refers to the service.

5. **Description of Service** provides a brief definition of the service.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 4-6.1 Service Parameters

Whereas each Common Service has fixed parameters, each Object–specific Service has unique parameters. This section defines those parameters for each Object–specific Service.

**Table 4-6.3 Object-specific Service Parameter Field Descriptions**

| Name | Type | Description of Request Parameters | Semantics of Values |
|---|---|---|---|
| 1 | 2 | 3 | 4 |

1. **Name** refers to the service request parameter.
2. **Type** specifies the data type of the service request parameter.
3. **Description of Request Parameters** describes the purpose of the request parameter.
4. **Semantics of Values** specifies the meaning of the values of the service request parameter, such as “the value is counts of microseconds.”

### 4-6.2 Service Response Data

Table 4-6.4 is the description of the service response data that includes the following:

**Table 4-6.4 Object-specific Service Response Data**

| Name | Type | Description of Response Data | Semantics of Values |
|---|---|---|---|
| 1 | 2 | 3 | 4 |

1. **Name** refers to the service response data.
2. **Type** specifies the data type of the service response data.
3. **Description of Response Data** describes the purpose of the response data.
4. **Semantics of Values** specifies the meaning of the values of the service response data.

This object definition component also includes a description of the usage and purpose of the service. If the service triggers a complex behavior, then you must specify it. Response Data from Common Services shall conform to that specified in Appendix A unless specified otherwise in the Object Definition or Device Profile.

## 4-7 Behavior

Behavior of an object may be triggered by an object’s services and is based on the values of the attributes accessed by the service. Together, the services and attribute values initiate state changes in the object. The behavior definition determines *how* an object responds when it receives notification of an event that changes its state. Behavior must be defined in terms of:

- the *state* an object is in when it receives notification of a state-changing event; and
- the *event* the object receives.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 4-7.1 State

A ***state*** is an object's current active mode of operation (e.,g., Running, Idle).

To define behavior in these terms, use a State Event Matrix and a State Transition Diagram when applicable. A ***State Event Matrix*** is a table that lists all possible events and services that initiate a state change, and indicates an object's response to the event or service based on the state of the object when it receives notification of that event.

For example, the table below shows three states (other objects may or may not have other states):

- ***Non–existent:*** the object has not yet been created. You must implement the create service to transition the object to existent.
- ***Idle:*** the object accepts services (e.g., Start, Stop, Get_Attribute_Single), but does not produce.
- ***Running:*** the object is performing its function.

**Table 4-7.1 State Event Matrix Table Format**

| Event | State | | |
|---|---|---|---|
| | **Non-existent** | **Idle** | **Running** |
| Create | Transition to Idle | Error: Object already exists. | Error: Object already exists. |
| Delete | Error: Object does not exist. (General Error Code 0x16) | Transition to Non-Existent | Error: Object State Conflict (General Error Code 0x0C) |
| Start | Error: Object does not exist. (General Error Code 0x16) | Transition to Running | Error: Object State Conflict (General Error Code 0x0C) |
| Stop | Error: Object does not exist. (General Error Code 0x16) | Error: Object State Conflict (General Error Code 0x0C) | Transition to Idle |
| Get_Attribute_Single | Error: Object does not exist. (General Error Code 0x16) | Validate/service the request. Return response | Validate/service the request. Return response |
| Set_Attribute_Single | Error: Object does not exist. (General Error Code 0x16) | Validate/service the request. Return response | Error: Object State Conflict (General Error Code 0x0C) |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 4-7.2 Event

An ***event*** is an external stimulus that could cause a state transition. A ***State Transition Diagram*** graphically illustrates the state of an object and includes services and attributes that cause it to transition to another state.

**Figure 4-7.1 State Transition Diagram Example**

When applicable, some object behavior definitions include an ***Attribute Access Table***, which lists all object attributes and how to access them in a given state.

**Table 4-7.2 Attribute Access Table Example**

| Attribute | State | | |
|---|---|---|---|
| | **Non-existent** | **Idle** | **Running** |
| Number_of_Members | Not available | Read Only | Read Only |
| Member_List | Not available | Read Only | Read Only |
| Data | Not available | Read/Write | Read Only |
| Owner | Not available | Read Only | Read Only |

## 4-8 Accessing Application object data

An Object Class definition must indicate any special provisions that have been made with regards to the external access of its associated data.

As described in Chapter 1, two types of Connections exist; Implicit (I/O) and Explicit Messaging. This section provides an overview of how these Connection types relate to Application Objects and how they can be used to access Application Object data. Figure 4-8.1 presents an overview of the relationship between the Connection Class and Application Objects and serves as a prefix to the overview.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 4-8.1 Connection Paths**

## 4-8.1 Access Through Explicit Messaging Connections

An Explicit Messaging Connection is capable of accessing any externally accessible object, including Application Objects. Explicit Messages can be used to deliver various service requests to Application Objects, including the reading and writing of Application Object attributes.

When an Explicit Message is received, the Explicit Messaging Connection delivers the message information to the Message Router. The Message Router delivers the message information to the appropriate *handler*. In the case of an Explicit Request Message, the *handler* is the target object specified in the request. In the case of an Explicit Response Message, the *handler* is the Client Application that previously issued the associated request.

## 4-8.2 Access Through I/O Connections

I/O Connections contain attributes that *point to* or reference the Application Object(s) to which they deliver received data and/or from which they obtain data to transmit.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 4-8.2 Access Paths to Application Object Data via I/O Connections**

Application Objects can be *directly* referenced by an I/O Connection. This is illustrated by **EXAMPLE 1** in Figure 4-8.2. In this example, Attribute #3 within Application Object Class #8/Instance #2 (*Att #3*), is being directly accessed by an I/O Connection. If this reference was made by the **produced_connection_path** attribute of the I/O Connection, then *Att #3* data would be the information being produced by the I/O Connection. If this reference was made by the **consumed_connection_path** attribute of the I/O Connection, then *Att #3* would be the recipient of the data that is consumed by the I/O Connection.

It is possible to access multiple attributes over a single I/O Connection. This is accomplished using a special Application Object called the *Assembly* Object. With respect to data exchange over an I/O Connection, an Assembly Object performs the following:

- Assembles separate pieces of Class, Instance, and/or Attribute data together so they can be produced by a single I/O Connection.
- Receives data from an I/O Connection that is associated with separate Classes, Instances, and/or Attributes and distributes the data accordingly.

With this in mind, Assembly Objects provide an *indirect* reference to various data items. **EXAMPLE 2** in Figure 4-8.2 illustrates an Assembly Object that groups Attributes #1 and #2 of Class #8/Instance #2 so they can be accessed by a single I/O Connection. Note that an Assembly Object is capable of incorporating attributes from more than 1 Object Class. See Chapter 5A–5 for a detailed definition of the Assembly Class.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 4-9 Extending Existing Objects and Defining New Objects

If your application requires a function that existing objects do not implement, you can:

- Add vendor–specific attributes to existing objects, or propose a standardized open addition to existing objects;
- Define a new open object or a new vendor–specific object (published or unpublished) to accommodate a new or revised functionality

Use the decision table below to determine how you would like to add functionality to your device.

An Object Class definition must indicate any special provisions that have been made with regards to the external access of its associated data.

**Table 4-9.1 How to Add Functionality**

| If you | And | Then |
|---|---|---|
| want to add attributes and/or services to an existing object to extend its functionality, | you want this extension to be standardized, | propose an Open Extension. |
| want to add attributes and/or services to an existing object to extend its functionality, | you want to provide user–access to the extension, | create and publish a Vendor–specific Extension. |
| want to add attributes and/or services to an existing object to extend its functionality, | you want to keep the extension proprietary, | create, but do NOT publish Vendor–specific Extension. |
| want to create a device that has a function outside the realm of existing objects, | you want this extension to be standardized, | propose a new Open Object. |
| want to create a device that has a function outside the realm of existing objects, | you want to provide user–access to the extension, | create and publish a new Vendor–specific Object. |
| want to create a device that has a function outside the realm of existing objects, | you want to keep the new object proprietary, | create but do NOT publish a new Vendor–specific Object. |

After determining whether you want to extend an existing object or define a new one, refer to the following pages for the appropriate information.

**Table 4-9.2 Where To Go To Make Changes**

| For information about: | Go to section: |
|---|---|
| CIP Object Address Ranges | 4-9.1 |
| Making Extensions to Objects | 4-9.2 |
| Vendor–specific Extensions | 4-9.2.1 |
| Open Extensions | 4-9.2.2 |
| Defining a New Object | 4-9.3 |
| Vendor–specific Object | 4-9.3.1 |
| Open Object | 4-9.3.2 |
| Defining a New Common Service | 4-9.4 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 4-9.1 CIP Object Address Ranges

Whether you decide to extend existing open objects or define new objects, you must know the CIP–defined Object Addressing Ranges. There are three categories:

**Table 4-9.3 Object Address Range Categories**

| This range | Refers to |
|---|---|
| Open | Open values are defined in the various volumes of the CIP Networks Library<br>• Chapter 5 has the complete list of open object classes defined thus far<br>• Open Attribute IDs are found in each of the individual object definitions<br>• Appendix A has the complete list of Common Services defined thus far |
| Vendor Specific | A range of values specific to the vendor of a device. These are used by vendors to extend their devices beyond the available *Open* options. A vendor internally manages the use of values within this range. Applies to object classes, attributes, and services. |
| Object Class Specific | A range of values who's meaning is defined by an Object Class. This range applies to Service Code definitions. |

Table 4-9.4 defines the ranges applicable to Class ID values.

**Table 4-9.4 Class ID**

| Range | Meaning | Quantity |
|---|---|---|
| 00 - $63_{hex}$ | Open | 100 |
| $64_{hex}$ - $C7_{hex}$ | Vendor Specific | 100 |
| $C8_{hex}$ - $EF_{hex}$ | Reserved by ODVA for future use | 40 |
| $F0_{hex}$ - $2FF_{hex}$ | Open | 528 |
| $300_{hex}$ - $4FF_{hex}$ | Vendor Specific | 512 |
| $500_{hex}$ - $FFFF_{hex}$ | Reserved by ODVA for future use | 64,256 |

Table 4-9.5 defines the ranges applicable to Attribute ID values.

**Table 4-9.5 Attribute ID Range**

| Range | Meaning | Quantity |
|---|---|---|
| 00 - $63_{hex}$ | Open | 100 |
| $64_{hex}$ - $C7_{hex}$ | Vendor Specific | 100 |
| $C8_{hex}$ - $FF_{hex}$ | Reserved by ODVA for future use | 56 |
| $100_{hex}$ – $2FF_{hex}$ | Open | 512 |
| $300_{hex}$ – $4FF_{hex}$ | Vendor Specific | 512 |
| $500_{hex}$ – $8FF_{hex}$ | Open | 1024 |
| $900_{hex}$ - $CFF_{hex}$ | Vendor Specific | 1024 |
| $D00_{hex}$ - $FFFF_{hex}$ | Reserved by ODVA for future use | 62208 |

Table 4-9.6 defines the ranges applicable to Service Code values.

**Table 4-9.6 Service Code Ranges**

| Range | Meaning | Quantity |
|---|---|---|
| 00 - $31_{hex}$ | Open. These are referred to as *Common Services*. These are defined in Appendix A. | 50 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Range | Meaning | Quantity |
|---|---|---|
| $32_{hex}$ - $4A_{hex}$ | Vendor Specific | 25 |
| $4B_{hex}$ - $63_{hex}$ | Object Class Specific | 25 |
| $64_{hex}$ - $7F_{hex}$ | Reserved by ODVA for future use | 28 |
| $80_{hex}$ - $FF_{hex}$ | Reserved for use as Reply Service Code (see Message Router Response Format in Chapter 2) | 128 |

## 4-9.2 Making Extensions to Open Objects

If an existing open object provides functions that closely match those of your product but do not completely support an implementation, then you may want to extend the existing object instead of creating a new object.

You can modify the existing object to include a new behavior that would satisfy your desired function. A modification could entail simply adding state flags or diagnostics, or could mean inheriting behaviors of another existing object class in your product line.

There are two types of extensions to open objects:

- vendor specific
  - published (via user manual, EDS, etc)
  - unpublished
- open (extensions may be made only via actions in CIP Special Interest Groups)

An extension of an existing open object definition:

- can add new open or vendor specific attributes to the most recent definition. Deletion of open attributes is not allowed. Open attributes can be obsoleted, but obsoleted attributes shall still be shown in the object definition and marked as obsolete.
- can only add open object attribute data to the end of the Get_ Attributes_ All and the Set_ Attributes_ All structure definitions via actions in CIP Special Interest Groups.
- when extending an object, if the behavior of the existing public attributes or public services will change or a new required attribute or service is added, the public revision of the object shall be incremented.
- can add open or vendor specific services to the list of services. Deletion of open services is not allowed. Open services can be obsoleted, but obsoleted services shall still be shown in the object definition and marked as obsolete.

### 4-9.2.1 Vendor–specific Extensions

If you want to add attributes and/or services to an open object (see Table 5A-1.1) to extend its functionality, and you want only your users to have access to the extension, then create a vendor–specific extension.

**Important:** Be sure to publish your vendor–specific extension in some way for your users. If you don’t publish the extension, then it is of little value because your users cannot access the added capability produced by the extension.

If you want to propose that your extension be standardized, then propose an open extension (see Section 4-9.2.2).

Regardless of whether you choose to keep the extension vendor–specific or propose it as an open extension, the process for defining an extension is the same.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Vendor specific extensions to open objects shall use the Vendor Specific ranges for Attribute IDs (see Table 4-9.5) and Service Codes (see Table 4-9.6).

Use the following example to examine the necessary steps for creating a vendor–specific, extension.

#### 4-9.2.1.1 Example

For example purposes, assume you have a *Widget Object*, which is an existing open object that has two Class attributes and three Instance Attributes.

**Table 4-9.7 Class Attributes**

| Number | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|
| 1 | Get | Revision | UINT | Revision of this object | |
| 2 | Get | Max Instance | UINT | Maximum Instance Number | |

**Table 4-9.8 Instance Attributes**

| Number | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|
| 1 | Get | Number of attributes | USINT | Number supported in this product | |
| 2 | Set | Output Value | BOOL | | 0 = off; 1 = on |
| 3 | Get | Status | UINT | Current status of this object instance | |

This object supports these Common Services:

- Get_Attributes_All
- Set_Attributes_All

These services specify the following attributes:

**Table 4-9.9 Attributes Specified by Object Services**

| Service | Attributes |
|---|---|
| Get_Attributes_All | 1, 2, 3 |
| Set_Attributes_All | 2 |

In addition to Common Services, the Widget Object in this example supports these Object Specific Services:

- Upload_Widget_Attributes_All
- Download_Widget_Attributes_All

The current capability of the Widget Object closely matches the function of your product. However, it does not completely support your desired implementation. You want to extend the object and then publish the extension for your users.

For example, this hypothetical device requires the instance attributes Output Value, Status, and Number of Attributes, but needs an additional attribute specifying *Input Value*, which you would like to publish to your users.

To extend the existing Widget Object to include the proprietary *Input Value* attribute:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

1. ***Add an attribute(s):*** Because this is a vendor specific attribute it needs to be in the vendor specific attribute value range (see Table 4-9.5). In this example, 100 has been chosen as the attribute number. The new list of Instance Attributes looks like this:

**Table 4-9.10 Instance Attributes for Extended Widget Object**

| Number | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|
| 1 | Get | Number of attributes | USINT | Number supported in this product | |
| 2 | Set | Output Value | BOOL | | 0 = off; 1 = on |
| 3 | Get | Status | UINT | Current status of this object instance | |
| **100** | **Set** | **Input Value** | **BOOL** | | **0 = off; 1 = on** |

Note that per section 4-5.1 and 4-5.2, vendor specific attributes shall not be included in Set_Attributes_All and Get_Attributes_All parameters.

2. ***Add one or more vendor specific service(s)***, if necessary, selecting a service code from the Vendor Specific range (see Table 4-9.6). Your vendor specific service may be defined to be specific to the object or common to all of your products.
3. ***Specify the new behavior*** with a new version of the State Transition Diagram and necessary text.

### 4-9.2.1.2 Implementing a Vendor–specific Extension

Whether adding vendor–specific attributes that are published for your users or proposed as a standard, you must plan for differences in:

- Revisions
- Get_Attributes_All service response
- Set_Attributes_All service request

#### 4-9.2.1.2.1 Revisions

Every object has a Class Attribute called “Revision.” The Revision of an object specifies the interface to that object, which encompasses all of the items in the object specification, including services, attributes, connections and behavior.

If the value of the Revision attribute is 01, then support of the Revision attribute is OPTIONAL in the object’s implementation. If the value is greater than 01, then support of the Revision attribute is REQUIRED.

**Table 4-9.11 Class Attributes**

| Number | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|
| 1 | Get | Revision | UINT | Revision of this object | |
| 2 | Get | Max Instance | UINT | Maximum Instance Number | |

**Important:** If you extend an object in a Server device but not in a potential Client, then the Revision of the object in the Server device is different from the revision expected by the Client. As a result, the Client will not recognize the extension, and will not implement its new behavior.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

To remedy this situation, you must document differences between the original object definition and the revised object definition. You may either:

- Update the Client; or
- Provide for the difference in revisions by allowing the Client to process error codes from the Server. You can program the Client to do one of the following:
  - Report the revision mismatch with an error handler that verifies the object revision; or
  - Determine the revision of the Server's object with an error handler that also performs special-case processing.

**Important:** The Revision class attribute is not the revision of an implementation (which is reflected in the Identity Object, Minor/Major revision status bits), but the revision of the **open object definition**.

If you want to specify a revision of the vendor–specific object definition, we recommend defining a "vendor–specific revision" class attribute in the vendor–specific address range.

The unrecognized attributes are either *critical* or *non–critical* to the behavior of the object.

**Table 4-9.12 Unrecognized Attribute Handling**

| If the unrecognized attributes are | Then |
|---|---|
| Critical | Processing the service results in an error. Client must be prepared to process the error. |
| Non–critical | the Server processes only the open attributes and ignores the vendor–specific attributes (which are initialized to default values). |

### 4-9.2.2 Open Extensions

If you want to add attributes and/or services to an existing open object to extend its functionality, and you want this extension to be standardized, then propose an open extension as follows:

1. Propose any new attributes using the Open ranges for Attribute IDs (see Table 4-9.5).
2. Propose the use of any additional or new common services (see Appendix A).
3. Propose any new object specific services using the Object Class Specific range for Service Codes (see Table 4-9.6).
4. Update the State Transition Diagram if necessary.
5. Consider whether the proposed changes will require incrementing the public revision (class attribute 1, Revision) of the object, as described in section 4-9.2 above.
6. Submit the extension to ODVA for approval.

**Important:** You cannot ship product based on your proposed extension until it is approved.

## 4-9.3 Defining a New Object

When defining a new object, either vendor–specific or open, follow these general guidelines:

- **Break large objects into smaller objects:** For example, break a "Controller" into "Data Table", which can be read and written, and "Program", which can be edited.
- **Avoid multiple interdependencies:** Don't break an object into multiple objects if this creates many interdependencies. Keep things that are tightly coupled in one object.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- **Hide some information:** Hide those aspects that are not important to the object being specified. For example, if you can store a set of data as an array or in a linked list, don't specify one or the other. Leave the set of data as "data storage".
- **Do not hide important semantics:** Don't hide aspects that are important to the object being specified. For example, don't write a value to a special location to cause a mode change. Use an interface that allows you to clearly state the desired action.
- **Generalize:** Ask, "If product A needs an object from product B, will it also be needed in product C? If this is a possibility, how can I define the object so that it is also useful in product C?"
- **Provide for expansion:** When possible, use variable length fields. You can extend fields in the future to handle cases not considered today.

  **Important:** It is preferable to extend an existing object than to define a new one. Fewer larger objects are more convenient than a proliferation of smaller objects that have similarities.

  In addition, use the numeric value itself instead of encoding the numeric value into a smaller number of bits, so that "unencodable" values become necessary in the future.

- **Be clear:** Keep all statements, references and terms as clear and obvious as possible by using meaningful names. Don't add complexity where not needed.
- **Make an object widely applicable:** Make sure the object is useful in a wide range of products (i.e., from a proximity sensor to a motor starter).

### 4-9.3.1 Vendor–specific

If you want your CIP–compatible product to perform a function that existing open objects cannot implement, then you may define a Vendor–specific Object identified by a class code in one of the Vendor Specific ranges within Table 4-9.4 Class ID. The object's attributes can be defined in any of the Open or Vendor Specific ranges defined in Table 4-9.5 Attribute ID Range, with the exception of Class attributes 1-7, which are defined in section 4-4.1. Your object may indicate that it supports any of the common services defined in Appendix A. In addition, it may define services in the Object Class Specific or Vendor Specific ranges of Table 4-9.6 Service Code Ranges. Your vendor specific services may be common to any of your vendor specific objects and/or your vendor specific extensions to open objects.

You may choose to create a vendor–specific object for one or more of the following reasons:

- You want to create a device that has a function outside the realm of existing objects.
- An existing product does not fit the function of existing objects.
- You want to create an object that has proprietary attributes and behavior.

Defining vendor–specific objects within specified Class ID, Attribute ID, and Service Code values has the following advantages and disadvantages:

**Table 4-9.13 Advantages and Disadvantages of Vendor-specific Objects**

| Advantages | Disadvantages |
|---|---|
| Vendor controls the object | The vendor must manage the dissemination of the object definition to other parties. |
| Innovation not stifled or slowed by standards process. | The Client must verify that the target device is of the vendor type that includes this new object definition. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 4-9.3.2 Open

If you think a new object and its behavior should be standardized, then propose an object specification to ODVA. ODVA will assign the class code from one of the open ranges (see Table 4-9.4) on acceptance. The proposed object must define its attributes in the Open ranges of Table 4-9.5. It must also specify which common services (see Appendix A) are used, and define any Object Class Specific services the new object will use (see Table 4-9.6).

You may follow the object template used in this chapter as well as the objects defined in Chapter 5, CIP Object Library. This template includes:

- Class code in open range
- Description of the object
- Class attributes
- Instance attributes
- Common services supported
- Object–specific services supported
- Connections
- Behavior

Submit the object proposal to one of the Special Interest Groups (SIGs) in ODVA for approval.

## 4-9.4 New Common Service

If you want to define a new service that extends the current list of Common Services, propose the service (in the open service code range) to ODVA.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This page is intentionally left blank


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Volume 1: Common Industrial Protocol (CIP™) Specification

# Chapter 5A: Object Library, Part A


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Contents

5A-1 Object Specifications ..... 14
5A-2 Identity Object ..... 20
5A-2.1 Class Attributes ..... 20
5A-2.2 Instance Attributes ..... 20
5A-2.2.1 Semantics ..... 22
5A-2.3 Common Services ..... 29
5A-2.3.1 Reset Service ..... 29
5A-2.3.2 Get_Attributes_All Response ..... 30
5A-2.4 Object–specific Services ..... 31
5A-2.5 Object-specific General and Extended Status Codes ..... 31
5A-2.6 Behavior ..... 33
5A-3 Message Router Object ..... 36
5A-3.1 Class Attributes ..... 36
5A-3.2 Instance Attributes ..... 36
5A-3.3 Common Services ..... 36
5A-3.3.1 Get_Attributes_All Response ..... 36
5A-3.4 Object–Specific Services ..... 37
5A-3.4.1 Symbolic Translation ..... 37
5A-3.5 Behavior ..... 38
5A-3.5.1 Service Request ..... 38
5A-3.5.2 Service Response ..... 39
5A-4 DeviceNet Object ..... 40
5A-5 Assembly Object ..... 41
5A-5.1 Class Attributes ..... 42
5A-5.2 Instance Attributes ..... 42
5A-5.3 Common Services ..... 42
5A-5.4 Object–specific Services ..... 43
5A-5.5 Behavior ..... 44
5A-5.5.1 Static Assemblies ..... 44
5A-5.5.2 Dynamic Assemblies ..... 45
5A-5.5.3 Connection Points ..... 46
5A-6 Connection Object ..... 47
5A-7 Connection Manager Object ..... 48
5A-8 Register Object ..... 49
5A-8.1 Class Attributes ..... 49
5A-8.2 Instance Attributes ..... 49
5A-8.3 Common Services ..... 49
5A-8.4 Object–specific Services ..... 49
5A-8.5 Behavior ..... 50
5A-9 Discrete Input Point Object ..... 51
5A-9.1 Revision History ..... 51
5A-9.2 Class Attributes ..... 51
5A-9.3 Instance Attributes ..... 52
5A-9.4 Common Services ..... 52
5A-9.4.1 Get_Attributes_All Response ..... 53
5A-9.4.2 Set_Attributes_All Request ..... 53
5A-9.5 Object–specific Services ..... 54
5A-9.6 Behavior ..... 54
5A-9.6.1 Attribute Access Rules ..... 56
5A-10 Discrete Output Point Object ..... 57
5A-10.1 Class Attributes ..... 57
5A-10.2 Instance Attributes ..... 57
5A-10.2.1 Value ..... 58


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

5A-10.2.2 Status ..... 59
5A-10.2.3 Fault and Idle Attributes ..... 59
5A-10.2.4 Run_Idle Command ..... 60
5A-10.2.5 Flash ..... 60
5A-10.2.6 Flash Rate ..... 60
5A-10.3 Common Services ..... 60
5A-10.3.1 Get_Attributes_All Response ..... 60
5A-10.3.2 Set_Attributes_All Request ..... 61
5A-10.4 Object–specific Services ..... 62
5A-10.5 Behavior ..... 62
5A-10.5.1 Attribute Access ..... 65
5A-11 Analog Input Point Object ..... 66
5A-11.1 Revision History ..... 66
5A-11.2 Class Attributes ..... 66
5A-11.3 Instance Attributes ..... 67
5A-11.4 Common Services ..... 68
5A-11.4.1 Get_Attributes_All Response ..... 68
5A-11.4.2 Set_Attributes_All Request ..... 69
5A-11.5 Object–specific Services ..... 69
5A-11.6 Behavior ..... 69
5A-11.6.1 Attribute Access Rules ..... 71
5A-12 Analog Output Point Object ..... 73
5A-12.1 Class Attributes ..... 73
5A-12.2 Instance Attributes ..... 73
5A-12.3 Common Services ..... 75
5A-12.3.1 Get_Attributes_All Response ..... 76
5A-12.3.2 Set_Attributes_All Request ..... 77
5A-12.4 Object–specific Services ..... 77
5A-12.5 Behavior ..... 77
5A-12.5.1 Attribute Access Rules ..... 81
5A-13 Presence Sensing Object ..... 83
5A-13.1 Class Attributes ..... 83
5A-13.2 Instance Attributes ..... 83
5A-13.2.1 Semantics ..... 84
5A-13.3 Common Services ..... 85
5A-13.4 Object–specific Services ..... 85
5A-13.5 Behavior ..... 85
5A-14 Parameter Object ..... 87
5A-14.1 Class Attributes ..... 88
5A-14.1.1 Semantics ..... 88
5A-14.2 Instance Attributes ..... 90
5A-14.2.1 Semantics ..... 92
5A-14.3 Common Services ..... 98
5A-14.3.1 Get_Attributes_All Response ..... 98
5A-14.4 Object–specific Services ..... 101
5A-14.5 Behavior ..... 102
5A-15 Parameter Group Object ..... 103
5A-15.1 Class Attributes ..... 103
5A-15.1.1 Semantics ..... 103
5A-15.2 Instance Attributes, Abbreviated Static Set ..... 104
5A-15.3 Instance Attributes, Extended Static Set ..... 104
5A-15.4 Common Services ..... 105
5A-15.4.1 Get_Attributes_All Response ..... 105
5A-15.5 Object–specific Services ..... 106
5A-15.6 Behavior ..... 106
5A-16 Group Object ..... 107


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

5A-16.1 Class Attributes ..... 107
5A-16.2 Instance Attributes ..... 108
5A-16.3 Common Services ..... 108
5A-16.3.1 Get_Attributes_All Response ..... 108
5A-16.4 Object–specific Services ..... 109
5A-16.5 Behavior ..... 109
5A-16.5.1 Attribute Access Rules ..... 110
5A-17 Discrete Input Group Object ..... 111
5A-17.1 Class Attributes ..... 111
5A-17.2 Instance Attributes ..... 112
5A-17.3 Semantics ..... 112
5A-17.3.1 On_OffDelay, Off_OnDelay ..... 112
5A-17.4 Common Services ..... 113
5A-17.4.1 Get_Attributes_All Response ..... 113
5A-17.4.2 Set_Attributes_All Request ..... 114
5A-17.5 Object–specific Services ..... 114
5A-17.6 Behavior ..... 114
5A-17.6.1 Attribute Access Rules ..... 115
5A-18 Discrete Output Group Object ..... 116
5A-18.1 Class Attributes ..... 116
5A-18.2 Instance Attributes ..... 117
5A-18.3 Semantics ..... 117
5A-18.3.1 Command, Fault Action, Fault Value, Idle Action, Idle Value ..... 117
5A-18.4 Common Services ..... 118
5A-18.4.1 Get_Attributes_All Response ..... 118
5A-18.4.2 Set_Attributes_All Request ..... 119
5A-18.5 Object–specific Services ..... 119
5A-18.6 Behavior ..... 120
5A-18.6.1 Attribute Access Rules ..... 120
5A-19 Discrete Group Object ..... 121
5A-19.1 Class Attributes ..... 121
5A-19.2 Instance Attributes ..... 122
5A-19.3 Common Services ..... 122
5A-19.3.1 Get_Attributes_All Response ..... 122
5A-19.4 Object–specific Services ..... 123
5A-19.5 Behavior ..... 123
5A-19.5.1 Attribute Access Rules ..... 124
5A-20 Analog Input Group Object ..... 125
5A-20.1 Class Attributes ..... 125
5A-20.2 Instance Attributes ..... 126
5A-20.3 Semantics ..... 126
5A-20.3.1 Value Data Type ..... 126
5A-20.4 Common Services ..... 127
5A-20.4.1 Get_Attributes_All Response ..... 127
5A-20.4.2 Set_Attributes_All Request ..... 128
5A-20.5 Object–specific Services ..... 128
5A-20.6 Behavior ..... 128
5A-20.6.1 Attribute Access Rules ..... 129
5A-21 Analog Output Group Object ..... 131
5A-21.1 Class Attributes ..... 131
5A-21.2 Instance Attributes ..... 132
5A-21.3 Semantics ..... 133
5A-21.3.1 Command, Fault Action, Fault Value, Idle Action, Idle Value, Value Data Type ..... 133
5A-21.3.2 Value Data Type ..... 133
5A-21.4 Common Services ..... 133
5A-21.4.1 Get_Attributes_All Response ..... 134


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

5A-21.4.2 Set_Attributes_All Request ..... 135
5A-21.5 Object–specific Services ..... 135
5A-21.6 Behavior ..... 135
5A-21.6.1 Attribute Access Rules ..... 136
5A-22 Analog Group Object ..... 138
5A-22.1 Class Attributes ..... 138
5A-22.2 Instance Attributes ..... 139
5A-22.3 Common Services ..... 139
5A-22.3.1 Get_Attributes_All Response ..... 140
5A-22.4 Object–specific Services ..... 140
5A-22.5 Behavior ..... 140
5A-22.5.1 Attribute Access Rules ..... 141
5A-23 Position Sensor Object ..... 143
5A-23.1 Revision History ..... 143
5A-23.2 Class Attributes ..... 143
5A-23.3 Instance Attributes ..... 143
5A-23.4 Semantics ..... 147
5A-23.4.1 Position Value Unsigned - Attribute 3 ..... 147
5A-23.4.2 Position Value Signed - Attribute 10 ..... 147
5A-23.4.3 Value Bit Resolution – Attribute 5 ..... 147
5A-23.4.4 Zero Offset – Attribute 6 ..... 148
5A-23.4.5 Auto Zero – Attribute 9 ..... 148
5A-23.4.6 Position Sensor Type – Attribute 11 ..... 149
5A-23.4.7 Direction Counting Toggle – Attribute 12 ..... 149
5A-23.4.8 Commissioning Diagnostic Control – Attribute 13 ..... 149
5A-23.4.9 Scaling Function Control – Attribute 14 ..... 149
5A-23.4.10 Position Format – Attribute 15 ..... 150
5A-23.4.11 Measuring Units per Span – Attribute 16 ..... 150
5A-23.4.12 Total Measuring Range in Measuring Units – Attribute 17 ..... 150
5A-23.4.13 Position measuring increment – Attribute 18 ..... 150
5A-23.4.14 Preset Value – Attribute 19 ..... 151
5A-23.4.15 COS Delta – Attribute 20 ..... 151
5A-23.4.16 Position State Register, Position limits – Attributes 21 to 23 ..... 151
5A-23.4.17 Velocity Format– Attribute 25 ..... 151
5A-23.4.18 Velocity Resolution – Attribute 26 ..... 152
5A-23.4.19 Minimum Velocity / Maximum Velocity - Attribute 27 thru 28 ..... 152
5A-23.4.20 Acceleration Value - Attribute 29 ..... 152
5A-23.4.21 Acceleration Format – Attribute 30 ..... 152
5A-23.4.22 Acceleration Resolution – Attribute 31 ..... 152
5A-23.4.23 Minimum Acceleration / Maximum Acceleration - Attribute 32 thru 33 ..... 152
5A-23.4.24 Number of CAM Channels – Attribute 34 ..... 152
5A-23.4.25 CAM Channel State Register – Attribute 35 ..... 154
5A-23.4.26 CAM Channel Polarity Register – Attribute 36 ..... 154
5A-23.4.27 CAM Channel Enable Register – Attribute 37 ..... 154
5A-23.4.28 CAM Low Limit – Attribute 38 ..... 155
5A-23.4.29 CAM High Limit – Attribute 39 ..... 155
5A-23.4.30 CAM Hysteresis – Attribute 40 ..... 155
5A-23.4.31 Operating Status – Attribute 41 ..... 155
5A-23.4.32 Physical Resolution – Attribute 42 ..... 155
5A-23.4.33 Number of Spans – Attribute 43 ..... 155
5A-23.4.34 Alarms – Attribute 44 ..... 156
5A-23.4.35 Supported Alarms – Attribute 45 ..... 156
5A-23.4.36 Alarm Flag – Attribute 46 ..... 156
5A-23.4.37 Warnings – Attribute 47 ..... 156
5A-23.4.38 Supported Warnings – Attribute 48 ..... 157
5A-23.4.39 Warning flag – Attribute 49 ..... 157


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

5A-23.4.40 Operating Time – Attribute 50 ... 157
5A-23.4.41 Offset Value – Attribute 51 ... 157
5A-23.5 Common Services ... 158
5A-23.6 Object–specific Services ... 159
5A-23.7 Behavior ... 159
5A-24 Position Controller Supervisor Object ... 162
5A-24.1 Revision History ... 162
5A-24.2 Class Attributes ... 162
5A-24.2.1 Consumed Axis Selection Number ... 162
5A-24.2.2 Produced Axis Selection Number ... 162
5A-24.3 Object Instance Attributes ... 163
5A-24.3.1 Supervisor Attributes ... 163
5A-24.3.2 Home and Index Attributes ... 164
5A-24.3.3 Registration Attributes ... 165
5A-24.3.4 Axis Following Attributes ... 166
5A-24.4 Common Services ... 166
5A-24.5 Object-specific Services ... 166
5A-24.6 Behavior ... 166
5A-24.6.1 Position Controller Supervisor State Diagrams ... 166
5A-25 Position Controller Object ... 169
5A-25.1 Revision History ... 169
5A-25.2 Class Attributes ... 169
5A-25.3 Instance Attributes ... 169
5A-25.3.1 Semantics ... 173
5A-25.4 Common Services ... 174
5A-25.5 Object-specific Services ... 174
5A-25.6 Behavior ... 174
5A-25.6.1 Position Controller State Diagrams ... 174
5A-26 Block Sequencer Object ... 177
5A-26.1 Class Attributes ... 177
5A-26.2 Instance Attributes ... 177
5A-26.3 Common Services ... 177
5A-26.4 Object-specific Services ... 178
5A-26.5 Behavior ... 178
5A-26.5.1 Block Sequencer State Diagrams ... 178
5A-27 Command Block Object ... 179
5A-27.1 Class Attributes ... 179
5A-27.2 Attributes ... 179
5A-27.3 Command Specific Attribute Services ... 179
5A-27.3.1 Modify Attribute Command - 01 ... 179
5A-27.3.2 Wait Equals Command - 02 ... 180
5A-27.3.3 Conditional Link Greater Than Command - 03 ... 180
5A-27.3.4 Conditional Link Less Than Command - 04 ... 181
5A-27.3.5 Decrement Counter Command - 05 ... 182
5A-27.3.6 Delay Command - 06 ... 182
5A-27.3.7 Trajectory Command - 07 ... 182
5A-27.3.8 Trajectory Command and Wait - 08 ... 182
5A-27.3.9 Velocity Change Command - 09 ... 183
5A-27.3.10 Goto Home Command - 10 ... 183
5A-27.3.11 Goto Index Command - 11 ... 183
5A-27.3.12 Goto Registration Position Command - 12 ... 183
5A-27.3.13 Common Services ... 184
5A-28 Motor Data Object ... 185
5A-28.1 Class Attributes ... 185
5A-28.2 Instance Attributes ... 185
5A-28.2.1 Motor Type Specific Motor Data Instance Attributes ... 185


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

5A-28.3 Common Services ..... 188
5A-28.4 Object–specific Services ..... 188
5A-28.5 Behavior ..... 188
5A-29 Control Supervisor Object ..... 189
5A-29.1 Class Attributes ..... 189
5A-29.2 Instance Attributes ..... 189
5A-29.2.1 Operation Mode Values ..... 191
5A-29.2.2 NetFaultMode Attribute ..... 191
5A-29.2.3 NetIdleMode Attribute ..... 191
5A-29.2.4 ProtectMode Attribute ..... 192
5A-29.2.5 Fault/Warning Code Style Attribute ..... 192
5A-29.3 Common Services ..... 193
5A-29.4 Object Specific Services ..... 193
5A-29.5 Behavior ..... 193
5A-29.5.1 Run/Stop Event Matrix ..... 194
5A-29.6 Fault and Warning codes ..... 195
5A-29.7 Object-specific General and Extended Status Codes ..... 202
5A-30 AC/DC Drive Object ..... 203
5A-30.1 Class Attributes ..... 203
5A-30.2 Instance Attributes ..... 203
5A-30.2.1 Semantics ..... 207
5A-30.3 Common Services ..... 208
5A-30.4 Object-specific Services ..... 208
5A-30.5 Behavior ..... 208
5A-30.5.1 Enhanced Speed Control Behavior ..... 211
5A-30.5.2 Scaling of attribute values ..... 211
5A-31 Acknowledge Handler Object ..... 213
5A-31.1 Class Attributes ..... 213
5A-31.2 Class Services ..... 213
5A-31.3 Instance Attributes ..... 214
5A-31.4 Instance Services ..... 215
5A-31.5 Object-specific Services ..... 215
5A-31.6 Behavior and Configuration of Acknowledged Data Production ..... 215
5A-31.6.1 Acknowledged Data Production ..... 216
5A-31.6.2 Change of State Examples ..... 216
5A-31.6.3 Use of Timers with Acknowledged Data Production ..... 218
5A-32 Overload Object ..... 223
5A-32.1 Class Attributes ..... 223
5A-32.2 Instance Attributes ..... 223
5A-32.2.1 Scaling of attribute values ..... 224
5A-32.3 Common Services ..... 224
5A-32.4 Object-specific Services ..... 224
5A-32.5 Behavior ..... 224
5A-33 Softstart Object ..... 225
5A-33.1 Class Attributes ..... 225
5A-33.2 Instance Attributes ..... 225
5A-33.3 Common Services ..... 226
5A-33.4 Object-specific Services ..... 226
5A-33.5 Behavior ..... 226
5A-33.5.1 Starting Behavior ..... 226
5A-33.5.2 Stopping Behavior ..... 227
5A-33.5.3 Other Attribute Behaviors ..... 228
5A-33.5.4 Soft Starter Output Voltage Behavior Description ..... 229
5A-34 Selection Object ..... 230
5A-34.1 Class Attributes ..... 230
5A-34.2 Class Services ..... 230


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

5A-34.3 Instance Attributes ..... 231
5A-34.4 Semantics ..... 232
5A-34.4.1 State ..... 232
5A-34.4.2 max_destinations ..... 232
5A-34.4.3 number_of_destinations ..... 232
5A-34.4.4 destination_list ..... 232
5A-34.4.5 max_sources ..... 232
5A-34.4.6 number_of_sources ..... 232
5A-34.4.7 source_list ..... 232
5A-34.4.8 source_used ..... 232
5A-34.4.9 source_alarm ..... 232
5A-34.4.10 algorithm_type ..... 233
5A-34.4.11 detection_count ..... 233
5A-34.4.12 selection_period ..... 233
5A-34.4.13 Object Source List ..... 233
5A-34.4.14 Destination Used ..... 233
5A-34.4.15 Input Data Value ..... 233
5A-34.4.16 Output Data Value ..... 233
5A-34.5 Instance Services ..... 234
5A-34.6 Instance Behavior ..... 234
5A-34.6.1 Message Distribution ..... 236
5A-34.6.2 Message Selection ..... 236
5A-34.6.3 Message Selection Algorithms ..... 237
5A-34.6.4 Examples ..... 242
5A-35 S-Device Supervisor Object ..... 243
5A-35.1 Class Attributes ..... 243
5A-35.1.1 Subclasses ..... 244
5A-35.2 Instance Attributes ..... 244
5A-35.2.1 Subclasses ..... 247
5A-35.3 Semantics ..... 248
5A-35.3.1 Device Type ..... 248
5A-35.3.2 Manufacturer’s Name ..... 248
5A-35.3.3 Software Revision Level ..... 248
5A-35.3.4 Hardware Revision Level ..... 248
5A-35.3.5 Manufacturer’s Serial Number ..... 248
5A-35.3.6 Device Status ..... 248
5A-35.3.7 Exception Status ..... 248
5A-35.3.8 Exception Detail Alarm and Exception Detail Warning ..... 249
5A-35.3.9 Common Exception Detail ..... 250
5A-35.3.10 Device Exception Detail ..... 250
5A-35.3.11 Manufacturer Exception Detail ..... 250
5A-35.3.12 Alarm Enable and Warning Enable ..... 251
5A-35.3.13 Time ..... 251
5A-35.3.14 Scheduled Maintenance Expiration Timer ..... 251
5A-35.3.15 Scheduled Maintenance Expiration Warning Enable ..... 252
5A-35.3.16 Endpoint ..... 252
5A-35.3.17 Recipe ..... 252
5A-35.4 Common Services ..... 252
5A-35.5 Object-specific Services ..... 253
5A-35.5.1 Abort ..... 253
5A-35.5.2 Recover ..... 253
5A-35.5.3 Perform_Diagnostics ..... 253
5A-35.5.4 TestID Parameter ..... 253
5A-35.6 Behavior ..... 254
5A-35.6.1 S-Device Supervisor Object States ..... 254
5A-35.6.2 State Event Matrix ..... 255


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

5A-35.6.3 S-Device Supervisor and Identity Object Interface ..... 256
5A-35.6.4 S-Device Supervisor and Application Object Interface ..... 256
5A-35.7 S-Device Supervisor Instance Subclass 01 — Power Generator ..... 256
5A-35.7.1 Instance Attributes ..... 256
5A-35.7.2 Services ..... 258
5A-35.7.3 Behavior ..... 258
5A-36 S-Analog Sensor Object ..... 259
5A-36.1 Class Attributes ..... 259
5A-36.1.1 Subclasses ..... 259
5A-36.2 Instance Attributes ..... 259
5A-36.2.1 Subclasses ..... 263
5A-36.3 Semantics ..... 264
5A-36.3.1 Data Type ..... 264
5A-36.3.2 Data Units ..... 264
5A-36.3.3 Value, Offset (A and B) and Gain ..... 264
5A-36.3.4 Status ..... 264
5A-36.3.5 Trip Points, Hysteresis and Settling Time ..... 264
5A-36.3.6 Safe State ..... 265
5A-36.3.7 Safe Value ..... 265
5A-36.3.8 Autozero Enable and Autozero Status ..... 265
5A-36.3.9 Autorange Enable and Range Multiplier ..... 265
5A-36.3.10 Produce Trigger Delta ..... 266
5A-36.3.11 Calibration Object Instance ..... 266
5A-36.3.12 Produce Trigger Delta Type ..... 266
5A-36.3.13 Value Descriptor ..... 266
5A-36.4 Common Services ..... 267
5A-36.5 Object–specific Services ..... 267
5A-36.5.1 Zero_Adjust and Gain_Adjust Services ..... 267
5A-36.6 Behavior ..... 268
5A-36.6.1 Data Type ..... 268
5A-36.7 S-Analog Sensor Object Class Subclass 01 (Instance Selector) ..... 268
5A-36.7.1 Class Attributes ..... 269
5A-36.7.2 Semantics ..... 269
5A-36.7.3 Services ..... 269
5A-36.7.4 Behavior ..... 270
5A-36.8 S-Analog Sensor Object Instance Subclass 01 (Flow Diagnostics) ..... 270
5A-36.8.1 Instance Attributes ..... 270
5A-36.8.2 Services ..... 270
5A-36.8.3 Behavior ..... 270
5A-36.9 S-Analog Sensor Object Instance Subclass 02 (Heat Transfer Vacuum Gauge) ..... 271
5A-36.9.1 Instance Attributes ..... 271
5A-36.9.2 Semantics ..... 271
5A-36.9.3 Services ..... 272
5A-36.9.4 Behavior ..... 272
5A-36.10 S-Analog Sensor Object Instance Subclass 03 (Diaphragm Gauge) ..... 272
5A-36.10.1 Instance Attributes ..... 272
5A-36.10.2 Semantics ..... 273
5A-36.10.3 Services ..... 274
5A-36.10.4 Behavior ..... 274
5A-36.11 S-Analog Sensor Object Instance Subclass 04 (Cold Cathode Ion Gauge) ..... 274
5A-36.11.1 Instance Attributes ..... 274
5A-36.11.2 Semantics: ..... 275
5A-36.11.3 Services ..... 276
5A-36.11.4 Behavior ..... 277
5A-36.12 S-Analog Sensor Object Instance Subclass 05 (Hot Cathode Ion Gauge) ..... 278
5A-36.12.1 Instance Attributes ..... 278


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

5A-36.12.2 Semantics ..... 280
5A-36.12.3 Services ..... 283
5A-36.12.4 Behavior ..... 284
5A-36.12.5 S-Analog Sensor Object Sub-State for the Executing State ..... 285
5A-36.12.6 S-Analog Sensor Sub-State Event Matrix ..... 286
5A-36.13 S-Analog Sensor Object Instance Subclass 06 - Transfer Function ..... 287
5A-36.13.1 Instance Attributes ..... 287
5A-36.13.2 Semantics ..... 287
5A-37 Analog Actuator Object ..... 289
5A-37.1 Class Attributes ..... 289
5A-37.1.1 Subclasses ..... 289
5A-37.2 Instance Attributes ..... 289
5A-37.2.1 Subclasses ..... 291
5A-37.2.2 Semantics: ..... 292
5A-37.3 Common Services ..... 294
5A-37.4 Object–Specific Services ..... 294
5A-37.5 Behavior ..... 294
5A-37.5.1 Data Type ..... 295
5A-38 S-Single Stage Controller Object ..... 296
5A-38.1 Class Attributes ..... 296
5A-38.1.1 Subclasses ..... 296
5A-38.2 Instance Attributes ..... 296
5A-38.2.1 Subclasses ..... 298
5A-38.3 Semantics: ..... 298
5A-38.3.1 Data Type ..... 298
5A-38.3.2 Data Units ..... 299
5A-38.3.3 Setpoint, Process Variable and Control Variable ..... 299
5A-38.3.4 Status ..... 299
5A-38.3.5 Control Mode ..... 299
5A-38.3.6 Safe State ..... 300
5A-38.3.7 Safe Value ..... 300
5A-38.4 Common Services ..... 300
5A-38.5 Object–specific Services ..... 300
5A-38.6 Behavior ..... 300
5A-38.6.1 Alarm and Warning Exception Conditions ..... 301
5A-38.6.2 Ramp Rate ..... 301
5A-38.6.3 Data Type ..... 301
5A-38.6.4 Control ..... 302
5A-38.7 S-Single Stage Controller Object Instance Subclass 01 - (PID and Source Select) ..... 302
5A-38.7.1 Instance Attributes ..... 302
5A-38.7.2 Semantics ..... 303
5A-38.7.3 Services ..... 304
5A-38.7.4 Behavior ..... 305
5A-38.8 S-Single Stage Controller Instance Subclass 02 - DC Generator ..... 307
5A-38.8.1 Instance Attributes ..... 307
5A-38.8.2 Semantics ..... 308
5A-38.8.3 Services ..... 308
5A-38.8.4 Behavior ..... 308
5A-38.9 S-Single Stage Controller Instance Subclass 03 - RF Generator ..... 309
5A-38.9.1 Instance Attributes ..... 309
5A-38.9.2 Semantics ..... 309
5A-38.9.3 Services ..... 310
5A-38.9.4 Behavior ..... 310
5A-38.10 S-Single Stage Controller Instance Subclass 04 - Frequency Control ..... 310
5A-38.10.1 Instance Attributes ..... 311
5A-38.10.2 Semantics ..... 311


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

5A-38.10.3 Services ..... 311
5A-38.10.4 Behavior ..... 311
5A-39 S-Gas Calibration Object ..... 312
5A-39.1 Class Attributes ..... 312
5A-39.1.1 Subclasses ..... 313
5A-39.2 Instance Attributes ..... 313
5A-39.2.1 Subclasses ..... 314
5A-39.3 Semantics: ..... 314
5A-39.3.1 Gas Standard Number ..... 314
5A-39.3.2 Valid Sensor Instances ..... 314
5A-39.3.3 Gas Symbol ..... 315
5A-39.3.4 Full Scale ..... 315
5A-39.3.5 Instance Application Example ..... 315
5A-39.4 Common Services ..... 315
5A-39.5 Object–specific Services ..... 315
5A-39.6 S-Gas Calibration Object Behavior ..... 316
5A-39.7 S-Gas Calibration Object Instance Subclass 01 (Standard T & P) ..... 316
5A-39.7.1 Instance Attributes ..... 316
5A-39.8 Services ..... 316
5A-39.9 Behavior ..... 316
5A-40 Trip Point Object ..... 317
5A-40.1 Class Attributes ..... 317
5A-40.1.1 Subclasses ..... 317
5A-40.2 Instance Attributes ..... 318
5A-40.2.1 Subclasses ..... 320
5A-40.3 Semantics ..... 320
5A-40.3.1 High/Low Trip Point, Enable and Output Status ..... 320
5A-40.3.2 Hysteresis and Delay ..... 320
5A-40.3.3 Data Type and Data Units ..... 321
5A-40.3.4 Polarity and Output ..... 321
5A-40.3.5 Source and Input ..... 321
5A-40.3.6 Destination and Output ..... 322
5A-40.4 Common Services ..... 322
5A-40.5 Object–Specific Services ..... 322
5A-40.6 Behavior ..... 322
5A-40.6.1 Trip Point Object States ..... 322
5A-40.6.2 Trip Point Object State Event Matrix ..... 323
5A-41 Drive Data Object ..... 324
5A-42 File Object ..... 325
5A-42.1 Class Attributes ..... 325
5A-42.2 Instance Attributes ..... 326
5A-42.2.1 Semantics ..... 327
5A-42.3 Common Services ..... 332
5A-42.3.1 Create Service Description ..... 332
5A-42.4 Object-specific Services ..... 333
5A-42.4.1 Initiate_Upload Service Description ..... 333
5A-42.4.2 Initiate_Download Service Description ..... 333
5A-42.4.3 Initiate_Partial_Read Service Description ..... 334
5A-42.4.4 Initiate_Partial_Write Service Description ..... 334
5A-42.4.5 Upload_Transfer Service Description ..... 335
5A-42.4.6 Download_Transfer Service Description ..... 336
5A-42.4.7 Clear_File Service Description ..... 337
5A-42.5 Error Codes ..... 338
5A-42.5.1 Error codes for Initiate Services ..... 338
5A-42.5.2 Error codes for Transfer Services ..... 338
5A-42.5.3 Error Codes for the Set_Attribute_Single Service ..... 339


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

5A-42.6 Behavior ..... 340
5A-42.7 Predefined File Instances ..... 344
5A-42.7.1 This Device's EDS File and Icon File – (0xC8) ..... 344
5A-42.7.2 Collection of Related EDS and Icon Files – (0xC9) ..... 345
5A-43 S-Partial Pressure Object ..... 347
5A-43.1 Class Attributes ..... 347
5A-43.1.1 Class Level Subclasses ..... 348
5A-43.1.2 Semantics ..... 348
5A-43.2 Instance Attributes ..... 349
5A-43.2.1 Instance Level Subclasses ..... 351
5A-43.2.2 Semantics ..... 351
5A-43.3 Common Services ..... 353
5A-43.3.1 Get_Attributes_All Response ..... 353
5A-43.4 Object–specific Services ..... 354
5A-43.4.1 Create_Range Service ..... 354
5A-43.4.2 Get_Instance_List Service ..... 355
5A-43.4.3 Get_Pressures Service ..... 355
5A-43.4.4 Get_All_Pressures Service ..... 355
5A-43.4.5 Group_Enable Service ..... 356
5A-43.5 Object Behavior ..... 356
5A-44 S-Sensor Calibration Object ..... 357
5A-44.1 Class Attributes ..... 357
5A-44.2 Subclasses ..... 357
5A-44.3 Instance Attributes ..... 358
5A-44.4 Subclasses ..... 359
5A-44.5 Semantics ..... 359
5A-44.5.1 Calibration ID Number ..... 359
5A-44.5.2 Valid Sensor Instances ..... 359
5A-44.5.3 Calibration Name ..... 359
5A-44.5.4 Full Scale ..... 360
5A-44.5.5 Temperature and Pressure ..... 360
5A-44.5.6 Instance Application Example ..... 360
5A-44.6 Common Services ..... 360
5A-44.7 Object–specific Services ..... 361
5A-44.7.1 Get_All_Instances Request Service Data Field Parameters ..... 361
5A-44.7.2 Get_All_Instances Success Response ..... 361
5A-44.8 S-Sensor Calibration Object Behavior ..... 361
5A-45 Event Log Object ..... 362
5A-45.1 Event Control ..... 362
5A-45.2 Class Attributes ..... 363
5A-45.2.1 Time Format ..... 363
5A-45.2.2 Present Time ..... 364
5A-45.2.3 Instance List ..... 364
5A-45.3 Instance Attributes ..... 364
5A-45.4 Semantics: ..... 366
5A-45.4.1 Attribute #1 - Instance Name ..... 366
5A-45.4.2 Attribute #2 - State ..... 366
5A-45.4.3 Attribute #3 - Event List Size ..... 366
5A-45.4.4 Attribute #4 - Event List ..... 366
5A-45.4.5 Attribute #5 - Event Enable ..... 366
5A-45.4.6 Attribute #6 - Event Silenced ..... 366
5A-45.4.7 Attribute #7 - Event State ..... 367
5A-45.4.8 Attribute #8 - Logged States Configuration ..... 367
5A-45.4.9 Attribute #9 - Logged Data Configuration ..... 367
5A-45.4.10 Attribute #10 - Log Full Action ..... 367
5A-45.4.11 Attribute #11 - Duplicate Event Action ..... 368


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

5A-45.4.12 Attribute #12 - Event/Data Log Maximum Size ..... 368
5A-45.4.13 Attribute #13 - Event/Data Log Size ..... 368
5A-45.4.14 Attribute #14 - Event/Data Log ..... 368
5A-45.4.15 Attribute #15 - Active Event List Size ..... 370
5A-45.4.16 Attribute #16 - Active Event List ..... 371
5A-45.4.17 Attribute #17 - Logged Event List Size ..... 371
5A-45.4.18 Attribute #18 - Logged Event List ..... 371
5A-45.4.19 Attribute #19 - Log Full ..... 371
5A-45.4.20 Attribute #20 - Log Not Empty ..... 371
5A-45.4.21 Attribute #21 - Log Overrun ..... 371
5A-45.4.22 Attribute #22 – Sequential Event/Data Access ..... 372
5A-45.4.23 Attribute #23 - Startup Behavior ..... 372
5A-45.4.24 Attribute #24 - Event Identifier Format ..... 373
5A-45.5 Common Services ..... 374
5A-45.5.1 Reset Service ..... 374
5A-45.6 Event/Data Log Instance Behavior ..... 375
5A-45.6.1 Event Behavior ..... 377


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-1 Object Specifications

Using the format previously defined, the objects listed in Table 5A-1.1 are specified in this object library. You can also use this table as a quick reference to objects and their corresponding class codes.

**Table 5A-1.1 Object Specifications in the CIP Object Library in Class Code Order**

| Class Code | Object Name | Go to page: | Associated Communications Object [1] |
|---|---|---|---|
| 01 hex | Identity Object | 5A-20 | No |
| 02 hex | Message Router Object | 5A-36 | No |
| 03 hex | DeviceNet Object | 5A-40 | Yes |
| 04 hex | Assembly Object | 5A-41 | No |
| 05 hex | Connection Object | 5A-47 | No |
| 06 hex | Connection Manager Object | 5A-48 | No |
| 07 hex | Register Object | 5A-49 | No |
| 08 hex | Discrete Input Point Object | 5A-51 | No |
| 09 hex | Discrete Output Point Object | 5A-57 | No |
| 0A hex | Analog Input Point Object | 5A-66 | No |
| 0B hex | Analog Output Point Object | 5A-73 | No |
| 0E hex | Presence Sensing Object | 5A-83 | No |
| 0F hex | Parameter Object | 5A-87 | No |
| 10 hex | Parameter Group Object | 5A-103 | No |
| 12 hex | Group Object | 5A-107 | No |
| 1D hex | Discrete Input Group Object | 5A-111 | No |
| 1E hex | Discrete Output Group Object | 5A-116 | No |
| 1F hex | Discrete Group Object | 5A-121 | No |
| 20 hex | Analog Input Group Object | 5A-125 | No |
| 21 hex | Analog Output Group Object | 5A-131 | No |
| 22 hex | Analog Group Object | 5A-138 | No |
| 23 hex | Position Sensor Object | 5A-143 | No |
| 24 hex | Position Controller Supervisor Object | 5A-162 | No |
| 25 hex | Position Controller Object | 5A-169 | No |
| 26 hex | Block Sequencer Object | 5A-177 | No |
| 27 hex | Command Block Object | 5A-179 | No |
| 28 hex | Motor Data Object | 5A-185 | No |
| 29 hex | Control Supervisor Object | 5A-189 | No |
| 2A hex | AC/DC Drive Object | 5A-202 | No |
| 2B hex | Acknowledge Handler Object | 5A-213 | No |
| 2C hex | Overload Object | 5A-223 | No |
| 2D hex | Softstart Object | 5A-225 | No |
| 2E hex | Selection Object | 5A-230 | No |
| 30 hex | S-Device Supervisor Object | 5A-242 | No |
| 31 hex | S-Analog Sensor Object | 5A-259 | No |
| 32 hex | S-Analog Actuator Object | 5A-289 | No |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Class Code | Object Name | Go to page: | Associated Communications Object [1] |
|---|---|---|---|
| 33 hex | S-Single Stage Controller Object | 5A-296 | No |
| 34 hex | S-Gas Calibration Object | 5A-312 | No |
| 35 hex | Trip Point Object | 5A-317 | No |
| N/A | Drive Data Object | 5A-324 | No |
| 37 hex | File Object | 5A-325 | No |
| 38 hex | S-Partial Pressure Object | 5A-347 | No |
| 39 hex | Safety Supervisor Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 3A hex | Safety Validator Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 3B hex | Safety Discrete Output Point Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 3C hex | Safety Discrete Output Group Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 3D hex | Safety Discrete Input Point Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 3E hex | Safety Discrete Input Group Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 3F hex | Safety Dual Channel Output Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 40 hex | S-Sensor Calibration Object | 5A-357 | No |
| 41 hex | Event Log Object | 5A-362 | No |
| 42 hex | Motion Device Axis Object | See Chapter 5 of Volume 9 | No |
| 43 hex | Time Sync Object | See Chapter 5B | No |
| 44 hex | Modbus Object | See Chapter 5B, also see Chapter 5 of Volume 7, Integration of Modbus Devices into the CIP Architecture | Yes |
| 45 hex | Originator Connection List Object | See Chapter 5B | No |
| 46 hex | Modbus Serial Link Object | See Chapter 5 of Volume 7, Integration of Modbus Devices into the CIP Architecture | Yes |
| 47 hex | Device Level Ring (DLR) Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | Yes |
| 48 hex | QoS Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | Yes |
| 49 hex | Safety Analog Input Point Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 4A hex | Safety Analog Input Group Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 4B hex | Safety Dual Channel Analog Input Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 4C hex | SERCOS III Link Object | See Chapter 5 of Volume 5, CIP Safety | Yes |
| 4D hex | Target Connection List Object | See Chapter 5B | No |
| 4E hex | Base Energy Object | See Chapter 5B | No |
| 4F hex | Electrical Energy Object | See Chapter 5B | No |
| 50 hex | Non-Electrical Energy Object | See Chapter 5B | No |
| 51 hex | Base Switch Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | No |
| 52 hex | SNMP Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | No |
| 53 hex | Power Management Object | See Chapter 5B | No |
| 54 hex | RSTP Bridge Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | Yes |
| 55 hex | RSTP Port Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | Yes |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Class Code | Object Name | Go to page: | Associated Communications Object [1] |
|---|---|---|---|
| 56 hex | Parallel Redundancy Protocol Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | Yes |
| 57 hex | PRP Nodes Table Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | Yes |
| 58 hex | Safety Feedback Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 59 hex | Safety Dual Channel Feedback Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 5A hex | Safety Stop Functions Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 5B hex | Safety Limit Functions Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 5C hex | Power Curtailment Object | See Chapter 5B | No |
| 5D hex | CIP Security Object | See Chapter 5 of Volume 8, CIP Security | No |
| 5E hex | EtherNet/IP Security Object | See Chapter 5 of Volume 8, CIP Security | No |
| 5F hex | Certificate Management Object | See Chapter 5 of Volume 8, CIP Security | No |
| F0 hex | ControlNet Object | See Chapter 5 of Volume 4, ControlNet Adaptation of CIP | Yes |
| F1 hex | ControlNet Keeper Object | See Chapter 5 of Volume 4, ControlNet Adaptation of CIP | Yes |
| F2 hex | ControlNet Scheduling Object | See Chapter 5 of Volume 4, ControlNet Adaptation of CIP | Yes |
| F3 hex | Connection Configuration Object | See Chapter 5B | No |
| F4 hex | Port Object | See Chapter 3 | No |
| F5 hex | TCP/IP Interface Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | Yes |
| F6 hex | Ethernet Link Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | Yes |
| F7 hex | CompoNet Link | See Chapter 5 of Volume 6, CompoNet Adaptation of CIP | Yes |
| F8 hex | CompoNet Repeater | See Chapter 5 of Volume 6, CompoNet Adaptation of CIP | Yes |

Table Footnotes:

1. Associated Communications Object is defined in the Port Object (instance attribute 11).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-1.2 Object Specifications in the CIP Object Library in "Object Name" Order**

| Class Code | Object Name | Go to page: | Associated Communications Object [1] |
|---|---|---|---|
| 2A hex | AC/DC Drive Object | 5A-202 | No |
| 2B hex | Acknowledge Handler Object | 5A-213 | No |
| 22 hex | Analog Group Object | 5A-138 | No |
| 20 hex | Analog Input Group Object | 5A-125 | No |
| 0A hex | Analog Input Point Object | 5A-66 | No |
| 21 hex | Analog Output Group Object | 5A-131 | No |
| 0B hex | Analog Output Point Object | 5A-73 | No |
| 04 hex | Assembly Object | 5A-41 | No |
| 4E hex | Base Energy Object | See Chapter 5B | No |
| 51 hex | Base Switch Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | No |
| 26 hex | Block Sequencer Object | 5A-177 | No |
| 5F hex | Certificate Management Object | See Chapter 5 of Volume 8, CIP Security | No |
| 5D hex | CIP Security Object | See Chapter 5 of Volume 8, CIP Security | No |
| 27 hex | Command Block Object | 5A-179 | No |
| F7 hex | CompoNet Link | See Chapter 5 of Volume 6, CompoNet Adaptation of CIP | Yes |
| F8 hex | CompoNet Repeater | See Chapter 5 of Volume 6, CompoNet Adaptation of CIP | Yes |
| F3 hex | Connection Configuration Object | See Chapter 5B | No |
| 06 hex | Connection Manager Object | 5A-48 | No |
| 05 hex | Connection Object | 5A-47 | No |
| 29 hex | Control Supervisor Object | 5A-189 | No |
| F1 hex | ControlNet Keeper Object | See Chapter 5 of Volume 4, ControlNet Adaptation of CIP | Yes |
| F0 hex | ControlNet Object | See Chapter 5 of Volume 4, ControlNet Adaptation of CIP | Yes |
| F2 hex | ControlNet Scheduling Object | See Chapter 5 of Volume 4, ControlNet Adaptation of CIP | Yes |
| 47 hex | Device Level Ring (DLR) Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | Yes |
| 03 hex | DeviceNet Object | 5A-40 | Yes |
| 1F hex | Discrete Group Object | 5A-121 | No |
| 1D hex | Discrete Input Group Object | 5A-111 | No |
| 08 hex | Discrete Input Point Object | 5A-51 | No |
| 1E hex | Discrete Output Group Object | 5A-116 | No |
| 09 hex | Discrete Output Point Object | 5A-57 | No |
| N/A | Drive Data Object | 5A-324 | No |
| 4F hex | Electrical Energy Object | See Chapter 5B | No |
| F6 hex | Ethernet Link Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | Yes |
| 5E hex | EtherNet/IP Security Object | See Chapter 5 of Volume 8, CIP Security | No |
| 41 hex | Event Log Object | 5A-362 | No |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Class Code | Object Name | Go to page: | Associated Communications Object [1] |
|---|---|---|---|
| 37 hex | File Object | 5A-325 | No |
| 12 hex | Group Object | 5A-107 | No |
| 01 hex | Identity Object | 5A-20 | No |
| 02 hex | Message Router Object | 5A-36 | No |
| 44 hex | Modbus Object | See Chapter 5B, also see Chapter 5 of Volume 7, Integration of Modbus Devices into the CIP Architecture | Yes |
| 46 hex | Modbus Serial Link Object | See Chapter 5 of Volume 7, Integration of Modbus Devices into the CIP Architecture | Yes |
| 42 hex | Motion Device Axis Object | See Chapter 5 of Volume 9 | No |
| 28 hex | Motor Data Object | 5A-185 | No |
| 50 hex | Non-Electrical Energy Object | See Chapter 5B | No |
| 45 hex | Originator Connection List Object | See Chapter 5B | No |
| 2C hex | Overload Object | 5A-223 | No |
| 56 hex | Parallel Redundancy Protocol Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | Yes |
| 10 hex | Parameter Group Object | 5A-103 | No |
| 0F hex | Parameter Object | 5A-87 | No |
| F4 hex | Port Object | See Chapter 3 | No |
| 25 hex | Position Controller Object | 5A-169 | No |
| 24 hex | Position Controller Supervisor Object | 5A-162 | No |
| 23 hex | Position Sensor Object | 5A-143 | No |
| 5C hex | Power Curtailment Object | See Chapter 5B | No |
| 53 hex | Power Management Object | See Chapter 5B | No |
| 0E hex | Presence Sensing Object | 5A-83 | No |
| 57 hex | PRP Nodes Table Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | Yes |
| 48 hex | QoS Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | Yes |
| 07 hex | Register Object | 5A-49 | No |
| 54 hex | RSTP Bridge Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | Yes |
| 55 hex | RSTP Port Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | Yes |
| 4A hex | Safety Analog Input Group Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 49 hex | Safety Analog Input Point Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 3E hex | Safety Discrete Input Group Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 3D hex | Safety Discrete Input Point Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 3C hex | Safety Discrete Output Group Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 3B hex | Safety Discrete Output Point Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 4B hex | Safety Dual Channel Analog Input Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 59 hex | Safety Dual Channel Feedback Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 3F hex | Safety Dual Channel Output Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 58 hex | Safety Feedback Object | See Chapter 5 of Volume 5, CIP Safety | No |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Class Code | Object Name | Go to page: | Associated Communications Object [1] |
|---|---|---|---|
| 5B hex | Safety Limit Functions Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 5A hex | Safety Stop Functions Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 39 hex | Safety Supervisor Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 3A hex | Safety Validator Object | See Chapter 5 of Volume 5, CIP Safety | No |
| 32 hex | S-Analog Actuator Object | 5A-289 | No |
| 31 hex | S-Analog Sensor Object | 5A-259 | No |
| 30 hex | S-Device Supervisor Object | 5A-242 | No |
| 2E hex | Selection Object | 5A-230 | No |
| 4C hex | SERCOS III Link Object | See Chapter 5 of Volume 5, CIP Safety | Yes |
| 34 hex | S-Gas Calibration Object | 5A-312 | No |
| 52 hex | SNMP Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | No |
| 2D hex | Softstart Object | 5A-225 | No |
| 38 hex | S-Partial Pressure Object | 5A-347 | No |
| 40 hex | S-Sensor Calibration Object | 5A-357 | No |
| 33 hex | S-Single Stage Controller Object | 5A-296 | No |
| 4D hex | Target Connection List Object | See Chapter 5B | No |
| F5 hex | TCP/IP Interface Object | See Chapter 5 of Volume 2, EtherNet/IP Adaptation of CIP | Yes |
| 43 hex | Time Sync Object | See Chapter 5B | No |
| 35 hex | Trip Point Object | 5A-317 | No |

Table Footnotes:

1. Associated Communications Object is defined in the Port Object (instance attribute 11).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-2 Identity Object

## Class Code: 01 Hex

This object provides identification of and general information about the device and optionally its subsystems. Instance one (1) of the Identity Object shall be present in all devices.

Instance one (1) shall identify the whole device. It alone shall be used for electronic keying, by applications wishing to determine what nodes are on the network and to match an EDS with a product on the network.

Other instances are optional. They may be provided by a device to give additional information about a device, such as its embedded components and/or subsystems.

Instances greater than one (1) that are used to report the revision of embedded components shall be assigned the Embedded Component (0xC8) device type. These instances exist to enable the manufacturer to report the revision of the embedded components of its choosing. See section 6-44 Embedded Component for attribute and service requirements for this purpose.

Instances greater than one (1) may also be used to identify internal subsystems that execute semi-independently. For example, a device may be designed to accept a 3rd party communications card.

## 5A-2.1 Class Attributes

**Table 5A-2.1 Identity Object Class Attributes**

| Number | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | This class attribute is optional and is described in Chapter 4 of this specification. | | | | | |
| 2 | Conditional[1] | Get | Max Instance | UINT | Maximum instance number of this object | The largest instance number of this object |
| 3 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |

1 If multiple instances of the Identity Object exist, then this attribute is required to be supported.

## 5A-2.2 Instance Attributes

**Table 5A-2.2 Identity Object Instance Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Required | Get | NV | Vendor ID | UINT | Identification of each vendor by number | See section 5A-2.2.1.1 |
| 2 | Required | Get | NV | Device Type | UINT | Indication of general type of product | See section 5A-2.2.1.2 |
| 3 | Required | Get | NV | Product Code | UINT | Identification of a particular product of an individual vendor | See section 5A-2.2.1.3 |
| 4 | Required | Get | NV | Revision | STRUCT of: | Revision of the item the Identity Object represents | See section 5A-2.2.1.4 |
| | | | | Major Revision | USINT | | |
| | | | | Minor Revision | USINT | | |
| 5 | Required | Get | V | Status | WORD | Summary status of device | See section 5A-2.2.1.5 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Identity Object, Class Code: $01_{Hex}$

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 6 | Required | Get | NV | Serial Number | UDINT | Serial number of device | See section 5A-2.2.1.6 |
| 7 | Required | Get | NV | Product Name | SHORT_ STRING | Human readable identification | See section 5A-2.2.1.7 |
| 8 | Optional | Get | V | State | USINT | Present state of the device as represented by the state transition diagram | 0 = Nonexistent<br>1 = Device Self Testing<br>2 = Standby<br>3 = Operational<br>4 = Major Recoverable Fault<br>5 = Major Unrecoverable Fault<br>6 – 254 = Reserved<br>255 = Default Value [1]<br>See section 5A-2.2.1.8 |
| 9 | Optional | Get | NV | Configuration Consistency Value | UINT | Contents identify configuration of device | See section 5A-2.2.1.9 |
| 10 | Optional | Get/Set | NV | Heartbeat Interval | USINT | The nominal interval between heartbeat messages in seconds. | The default value is 0. Zero disables transmission of the heartbeat message.<br>See section 5A-2.2.1.10 |
| 11 | Optional | Set | NV | Active Language | STRUCT of | Currently active language for the device. | Based on ISO 639-2/T. See Appendix C-4, STRINGI data type, language1, language2, and language3 fields. See section 5A-2.2.1.11 |
| | | | | | USINT | The language1 field from the STRINGI data type. | |
| | | | | | USINT | The language2 field from the STRINGI data type. | |
| | | | | | USINT | The language3 field from the STRINGI data type. | |
| 12 | Optional | Get | V | Supported Language List | ARRAY of STRUCT of | List of languages supported by character strings of data type STRINGI within the device. | See Appendix C-4, STRINGI data type, language1, language2, and language3 fields. The array index of the language is used within the International String Selection member protocol. |
| | | | | | USINT | The language1 field from the STRINGI data type. | |
| | | | | | USINT | The language2 field from the STRINGI data type. | |
| | | | | | USINT | The language3 field from the STRINGI data type. | |
| 13 | Optional | Get | NV | International Product Name | STRINGI | The names of the product in various languages | |
| 14 | Optional | Set | V | Semaphore | Struct of | Access Synchronization Semaphore | See section 5A-2.2.1.12<br>Default: All zero values<br>Timer valid range 100 thru 32767 |
| | | | | Client Electronic Key | UINT | Vendor Number | |
| | | | | | UDINT | Client Serial Number | |
| | | | | Semaphore Timer | ITIME | Millisecond Timer | |
| 15 | Optional | Set | NV | Assigned_Name | STRINGI | User assigned name. | See section 5A-2.2.1.13 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Identity Object, Class Code: $01_{Hex}$**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 16 | Optional | Set | NV | Assigned_Description | STRINGI | User assigned description. | See section 5A-2.2.1.14 |
| 17 | Optional | Set | NV | Geographic_Location | STRINGI | User assigned location. | See section 5A-2.2.1.15 |
| 18 | Conditional[2] | | | Modbus Identity Info | See Volume 7, Integration of Modbus Devices into the CIP Architecture (Chapter 5) | | |
| 19 | Optional | Get | V | Protection Mode | WORD | Current protection mode of the device | See section 5A-2.2.1.16 |

Table Footnotes:

1 The Default Value shall be used in the Get_Attributes_All response if the attribute is not implemented.

2 This attribute is not allowed if the device is not a Modbus device. If the device is a Modbus device, see Volume 7.

The following Instance Attributes are used to identify with certainty the appropriate device targeted by a connection originator:

- Vendor ID
- Device Type
- Product Code
- Revision

This collection of attributes, when kept by the connection originator, is often referred to as a device's "electronic key".

### 5A-2.2.1 Semantics

#### 5A-2.2.1.1 Vendor ID – Attribute 1

Vendor IDs are managed by ODVA. The value zero is not valid.

#### 5A-2.2.1.2 Device Type – Attribute 2

The Device Type value is used to identify the device profile (see Chapter 6, Device Profiles) that a particular product is using. Device profiles define minimum requirements a device must implement, as well as common options.

The list of public device profiles is managed by ODVA. A listing of the presently defined public device profiles and their device type values can be found in section 6-7.

The Embedded Component (0xC8) device type is not allowed for instance 1. See section 6-44, Embedded Component.

Device Type value zero has been deprecated. The Generic device profile was previously assigned the value zero, and devices do exist in the field with that Device Type number. See section 6-8 for more information.

#### 5A-2.2.1.3 Product Code – Attribute 3

The vendor assigned Product Code identifies a particular product within a device type. Each vendor assigns this code to each of its products. The Product Code typically maps to one or more catalog/model numbers. Products shall have different codes if their configuration and/or runtime options are different. Such devices present a different logical view to the network. On the other hand for example, two products that are the same except for their color or mounting feet are the same logically and may share the same product code.

The value zero is not valid.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-2.2.1.4 Revision – Attribute 4

The Revision attribute, which consists of Major and Minor Revisions, identifies the *Revision* of the item the Identity Object is representing.

The value zero is not valid for either the Major or Minor Revision fields. If the Device Type of the instance is Embedded Component (0xC8), the Major Revision and/or Minor Revision may be zero. See section 6-44, Embedded Component.

The **Major** and **Minor** Revision are typically displayed as major.minor. Minor revisions shall be displayed as three digits with leading zeros as necessary. The Major Revision is limited to values between 1 and 127; the eighth bit (representing, when set to one, values of 128 – 255) is reserved by CIP and shall have a value of zero.

The major revision should be incremented by the vendor when there is a significant change to the 'fit, form, or function' of the product. Any changes that affect the configuration choices available to the user (and therefore the Electronic Data Sheet) require incrementing the major revision.

The minor revision is typically used to identify changes in a product that do not affect user configuration choices. For example, bug fixes, hardware component change, labeling change, etc. Changes in minor revision are not used by a configuration tool to match a device with an Electronic Data Sheet.

### 5A-2.2.1.5 Status – Attribute 5

This attribute represents the current status of the entire device. Its value changes as the state of the device changes. The Status attribute is a WORD, with the following bit definitions:

**Table 5A-2.3 Bit Definitions for Status Instance Attribute of Identity Object**

| Bit (s) | Called | Definition |
|---|---|---|
| 0 | Owned | TRUE indicates the device (or an object within the device) has an owner. Within the Master/Slave paradigm the setting of this bit means that the Predefined Master/Slave Connection Set has been allocated to a master. Outside the Master/Slave paradigm the meaning of this bit is TBD. |
| 1 | | Reserved, shall be 0 |
| 2 | Configured | TRUE indicates the application of the device has been configured to do something different than the "out–of–box" default. This shall not include configuration of the communications. |
| 3 | | Reserved, shall be 0 |
| 4 – 7 | Extended Device Status | Vendor–specific or as defined by Table 5A-2.4. The EDS shall indicate if the device instance (1) follows a vendor-specific definition for these bits by using the DeviceStatusAssembly keyword as defined in section 7-3.6.3, Device Description Section. No mechanism is provided to enumerate the vendor-specific use of these values in the EDS file for instances greater than one (1). |
| 8 | Minor Recoverable Fault | TRUE indicates the device detected a problem with itself, which is thought to be recoverable. The problem does not cause the device to go into one of the faulted states. See Behavior section. |
| 9 | Minor Unrecoverable Fault | TRUE indicates the device detected a problem with itself, which is thought to be unrecoverable. The problem does not cause the device to go into one of the faulted states. See Behavior section. |
| 10 | Major Recoverable Fault | TRUE indicates the device detected a problem with itself, which caused the device to go into the "Major Recoverable Fault" state. See Behavior section. |
| 11 | Major Unrecoverable Fault | TRUE indicates the device detected a problem with itself, which caused the device to go into the "Major Unrecoverable Fault" state. See Behavior section. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Bit (s) | Called | Definition |
|---|---|---|
| 12 - 15 | Extended Device Status 2 | Reserved - (shall be 0) or vendor-specific. The EDS shall indicate if the device instance (1) follows a vendor-specific definition for these bits by using the DeviceStatusAssembly2 keyword as defined in Section 7-3.6.3, Device Description Section. No mechanism is provided to enumerate the vendor-specific use of these values in the EDS file for instances greater than one (1). |

**Table 5A-2.4 Default Values for Extended Device Status Field (bits 4-7) in Status Instance Attribute**

| Value | Description |
|---|---|
| 0 | Self-Testing or Unknown |
| 1 | Firmware Update in Progress |
| 2 | At least one faulted I/O connection |
| 3 | No I/O connections established |
| 4 | Non-Volatile Configuration bad |
| 5 | Major Fault – either bit 10 or bit 11 in Table 5A-2.3 is true (1) |
| 6 | At least one I/O connection in run mode |
| 7 | At least one I/O connection established, all in idle mode |
| 8 | The Status attribute is not applicable to this instance. Valid only for instances greater than one (1). |
| 9 | Reserved |
| 10 thru 15 | Vendor specific [1] |

Table Footnotes:

1 No mechanism is provided to enumerate the vendor-specific use of these values in the EDS file.

The values of the following ***Status*** bits indicate the state of the device:

- Bit 8: Minor Recoverable Fault
- Bit 9: Minor Unrecoverable Fault
- Bit 10: Major Recoverable Fault
- Bit 11: Major Unrecoverable Fault

Note that the events that constitute a fault (recoverable or unrecoverable) are to be determined by the product developer. The following examples should help to define the various types of faults:

- Minor Recoverable Fault - an analog input device is sensing an input that exceeds the configured maximum input value.
- Minor Unrecoverable Fault - the device's battery backed RAM requires a battery replacement. The device will continue to function properly until the first time power is cycled.
- Major Recoverable Fault - the device's configuration is incorrect or incomplete.
- Major Unrecoverable Fault - the device failed its ROM checksum process.

**Table 5A-2.5 Minor/Major Bit Definitions**

| | Recoverable | Non–recoverable |
|---|---|---|
| Minor (no state change) | Bit 8 | Bit 9 |
| Major (state changes) | Bit 10 | Bit 11 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The event that sets the bit also causes a state change. See the State Transition Diagram in Figure 5A-2.1.

#### 5A-2.2.1.6 Serial Number – Attribute 6

This attribute is a number used in conjunction with the Vendor ID to form a unique identifier for each device on any CIP network. Each vendor is responsible for guaranteeing the uniqueness of the serial number across all of its devices.

The value zero is not a valid serial number and shall not be assigned to a product nor reported by a compliant device. Reception of the value zero when accessing this attribute indicates the target is not a CIP device and does not contain a serial number.

#### 5A-2.2.1.7 Product Name – Attribute 7

This text string shall represent a short description of the product represented by the Product Code in attribute 3. The same Product Code may have a variety of Product Name strings. The maximum number of characters in this string is 32. Note: The logical view to the network (described in section 5A-2.2.1.3, Product Code) does not include the Product Name.

#### 5A-2.2.1.8 State – Attribute 8

This attribute is an indication of the present state of the device. Note that the nature of a Major Unrecoverable Fault could be such that it may not be accurately reflected by the State attribute.

#### 5A-2.2.1.9 Configuration Consistency Value – Attribute 9

A product may automatically modify the *Configuration Consistency Value* whenever any non-volatile attribute is altered. A client node may, or may not, compare this value to a value within its own memory prior to system operation. The client node's behavior, upon detection of a mismatch, is vendor specific. The *Configuration Consistency Value* may be a CRC, incrementing count or any other mechanism. The only requirement is that if the configuration changes, the *Configuration Consistency Value* shall be different to reflect the change.

#### 5A-2.2.1.10 Heartbeat Interval – Attribute 10

This attribute sets the nominal interval between production of optional heartbeat messages.

#### 5A-2.2.1.11 Active Language – Attribute 11

If this attribute is supported, all character strings within the device of data type STRING or SHORT_STRING shall follow this setting. Only languages, which are supported by the ISO-8859-1 character set, can be represented in these strings; if the Active Language is set to a language, which cannot be represented in ISO-8859-1, the device shall return the string in the default language. Any other object attributes, (class or instance level) which set the language for strings of these data types, shall not be supported (i.e. the class level Native Language attribute of the Parameter and Parameter Group objects).

#### 5A-2.2.1.12 Semaphore – Attribute 14

This optional attribute provides a semaphore for client access synchronization to the entire device. This is a volatile attribute whose value after a reset or power-up is always zero. This attribute consists of two components, an electronic key of the client process and a semaphore timer.

The Client Electronic Key is composed of the client's CIP vendor number and the client's device serial number.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The Semaphore Timer is a millisecond timer, that when non-zero, counts down to zero. When the Semaphore Timer reaches the value zero, the Semaphore attribute is set to all zeros. The valid range of values for the timer portion of this attribute in a Set_Attribute_Single's service data is 100 milliseconds through 32767 milliseconds.

The Semaphore attribute can be read at any time. The value reported will be the current content of the Semaphore attribute, including the real-time (actual) timer value.

The Semaphore attribute's Semaphore Timer component operates at the base time resolution of the device. Therefore, if the time base of the device is greater than a 1 millisecond resolution, the time value accepted by the attribute shall be rounded up to the next timer increment appropriate for the device.

When a Set_Attribute_Single service is directed to this attribute, the attribute will be set to the service data if either of the following is true:

- The current value in the attribute is zero
- The Electronic Key portion of the service data matches the Electronic Key portion of the attribute data

The following error responses are defined for the Semaphore attribute:

| Semaphore Attribute Error Responses | | |
|---|---|---|
| **Error Code** | **Additional Code** | **Error Response Interpretation** |
| 0x02 | None | Resource Unavailable – The Client Electronic Key portion of the Semaphore attribute was non-zero and did not match the Client Electronic Key portion of the Set Attribute service data. This error response indicates that the Semaphore has been previously set by a different client process and is not available at the present time. |
| 0x09 | None | Invalid Attribute Value – The value specified for the timer portion of the attribute value was less than 100 milliseconds. |
| 0x10 | None | Device State Conflict – the device is currently in a state that does not allow the setting of the Semaphore attribute. This is a device specific behavior and shall be documented by the vendor. |

### 5A-2.2.1.13 Assigned_Name – Attribute 15

This text string represents the user assigned name of the device.

### 5A-2.2.1.14 Assigned_Description – Attribute 16

This text string represents a user assigned description of the device and may also describe its function.

### 5A-2.2.1.15 Geographic_Location – Attribute 17

This text string represents the physical location of the device as provided by the user.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-2.2.1.16 Protection Mode – Attribute 19

This attribute is an indication of the present mode of protection for the device.

**Table 5A-2.6 Protection Mode Attribute Bit Definitions**

| Bit (s) | Called | Definition | |
|---|---|---|---|
| 0-2 | Implicit Protection Setting | Indicates the current implicit protection setting of the device. See section 5A-2.2.1.16.1 | Bit 0 = Implicit Protection (clear if Not Protected; set if Protected)<br>Bits 1-2 = Reserved for future use |
| 3 | Explicit Protection Setting | Indicates the current explicit setting of the device | Explict Protection (clear if Not Protected; set if Protected) |
| 4-15 | Reserved | Reserved for future use and shall be set to zero. | |

#### 5A-2.2.1.16.1 Protection Settings

Two protection settings are defined: Implicit Protection (bits 0-2) and Explicit Protection (bit 3). The default value for both the Implicit and Explicit Setting is Not Protected. When both the Implicit and Explicit Protection Settings are Not Protected, the device shall accept all CIP explicit messages that are valid for the device, subject to any object-specific or device-specific rules regarding state conflicts.

The Implicit Protection setting shall indicate that the device has entered a state in which it rejects explicit message requests that would disrupt its operation. The conditions under which a device enters the Implicit Protection setting are device-specific. The following examples, while not all inclusive, are conditions under which a device should consider entering the Implicit Protection setting:

1. Presence of an established target I/O connection. This method is highly recommended for devices whose primary means of control and/or data production is via an I/O connection. For O->T connections that include the Run/Idle header, it may be further qualified that the connection be in run mode.
2. The device is being controlled or run locally via a local user interface.
3. The device is being controlled via an explicit message connection.

NOTE: Devices shall exit the Implicit Protection setting when the conditions under which it entered the Implicit Protection setting are no longer present.

When in the Implicit Protection setting, the device shall reject those CIP explicit messages that would be disruptive to the device's current operation. The following lists the disruptive explicit message requests that a device shall reject when in Implicit Protection setting:

1. Reset service to the Identity Object.
2. Device firmware update.
3. Changes to communication settings that would cause loss of communications with the device.

Additional explicit message requests that a device may reject when in the Implicit Protection setting are device-specific. The items protected when in Implicit Protection setting are referred to as the "Implicit Protection Policy".

The Explicit Protection setting shall indicate that the device has been explicitly placed in a protected mode, either by hardware means or via a software-based setting.

The following list shows example methods for placing a device in Explicit Protection setting:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

1. Key switch on the device is placed in the "Run" position.
2. Rotary switch on the module is set to a specific value to enable Explicit Protection setting.
3. The device is placed in Explicit Protection setting via a local user interface (e.g., touch screen, keypad, etc.).
4. The device is placed in Explicit Protection setting via secure web server interface or other secure connection means.

When in the Explicit Protection setting the device shall reject CIP explicit message requests that would alter the device's configuration settings or otherwise be disruptive to the device's current operation. The device may provide a (vendor-specific) mechanism for configuration of which disruptive explicit messages are to be rejected when in Explicit Protection mode. This mechanism should only be available/accessible while the device is Not Protected.

A device that implements the Protection Mode attribute may implement the Implicit Protection setting, Explicit Protection setting, or both. When both settings are implemented, the Explicit Protection setting should by default include the explicit messages covered by the Implicit Protection setting. The Implicit and Explicit Protection settings may be active simultaneously, for example when the device has been placed in the Explicit Protection setting, and also has an I/O connection which causes disruptive services to be rejected. The following diagram illustrates the recommended behavior of a device that implements both the Implicit and Explicit Protection settings:

**Figure 5A-2.1 Explicit and Implicit Setting Interaction**

Devices shall publish in user documentation the conditions under which they enter/exit either of the protection settings and what items are covered by each level supported. The items protected when in Explicit Protection setting are referred to as the "Explicit Protection Policy".

Any explicit messages that are rejected when the device is in either of the protection levels shall be returned with a General Status code of 0x10 (Device State Conflict).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-2.3 Common Services

The Identity Object provides the following Common Services:

**Table 5A-2.7 Identity Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 01hex | Optional | Required | Get_Attributes_All | Returns a predefined listing of this objects attributes (See the Get_Attributes_All Response definition below) |
| 05hex | Optional | Required | Reset | Invokes the Reset service for the device. |
| 0Ehex | Conditional [1] | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 10hex | n/a | Conditional [2] | Set_Attribute_Single | Modifies an attribute (Required if Heartbeat interval is defined) |
| 11hex | Conditional [4] | n/a | Find_Next_Object_Instance | Causes the specified Class to search for and return a list of instance IDs of existing instances of the Identity object. |
| 18hex | Optional | Conditional [3] | Get_Member | Returns the content of a selected member of an attribute. |

Table Footnotes:

1 The Get_Attribute_Single service is Required if any Class attributes are implemented.
2 The Set_Attribute_Single service is Required if any settable attributes are implemented.
3 The Get_Member service is Required if any attributes with the International String (STRINGI) data type are implemented.
4 Required if non-consecutive instances exist, optional otherwise

See Appendix A for definition of these services

### 5A-2.3.1 Reset Service

When the Identity Object receives a Reset request, it:

- determines if it can provide the type of reset requested
- responds to the request
- attempts to perform the type of reset requested

The Reset common service has the following object–specific request parameter:

**Table 5A-2.8 Reset Service Request Parameters**

| Name | Type | Description of Request Parameters | Semantics of Values |
|---|---|---|---|
| Reset Type | USINT | Type of Reset. This parameter is optional. If this parameter is omitted, the default value of Power Cycle (0) is used. | See Table below. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The Reset Type parameter for the Reset common service has the following values defined:

**Table 5A-2.9 Reset Type Parameter Values**

| Value: | Need in Implementation | Name | Type of Reset: |
|---|---|---|---|
| 0 | Required | Power Cycle | Emulate as closely as possible cycling power on the item the *Identity Object* represents. This value is the default if this parameter is omitted. |
| 1 | Optional | Return to Factory Defaults | Return as closely as possible to the factory default configuration, then emulate cycling power as closely as possible. |
| 2 | Optional | Return to Factory Defaults except Communications Parameters | Return as closely as possible to the out-of-box configuration with the exception of communication link parameters and emulate cycling power as closely as possible. The communication link parameters that are to be preserved are defined by each network type. See the Reset service of the network specific link or Identity object(s) in the relevant network volume for complete information. |
| 3 – 99 | | | Reserved by CIP |
| 100 – 199 | | | Vendor specific |
| 200 - 255 | | | Reserved by CIP |

The Reset common service has no object-specific response parameters.

### 5A-2.3.2 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-2.10 Get_Attributes_All Response – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 |
| 2 | UINT | Max Instance | 1 |
| 6 | UINT | Max ID Number Class Attributes | 0 |
| 7 | UINT | Max ID Number Instance Attributes | 0 |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-2.11 Get_Attributes_All Response – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Vendor ID | |
| 2 | UINT | Device Type | |
| 3 | UINT | Product Code | |
| 4 | STRUCT of: | Revision | |
| | USINT | Major Revision | |
| | USINT | Minor Revision | |
| 5 | WORD | Status | |
| 6 | UDINT | Serial Number | |
| 7 | SHORT STRING | Product Name | |
| 8 | USINT | State | 255 |
| 9 | UINT | Configuration Consistency Value | 0 |
| 10 | USINT | Heartbeat Interval | 0 |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

**Important:** If the Get_Attributes_All service returns data after the Heartbeat Interval and the Identity Object Revision is 1, the data after the Heartbeat Interval shall be ignored (since previous editions of the specification contained unparsable content past that point).

## 5A-2.4 Object–specific Services

The Identity Object provides no Object–specific services.

## 5A-2.5 Object-specific General and Extended Status Codes

**Table 5A-2.12 Object-specific General and Extended Status Codes**

| General Status Code (in hex) | 8-bit Associated Extended Status Codes | Status Name | Description of Status |
|---|---|---|---|
| 00 - CF | | General Codes | Defined in Appendix B |
| | 00 – EE | | Reserved Extended Status Codes |
| | F0 - FE | Vendor Specific | Vendor specific Extended Codes |
| | FF | | Used with all General Codes when required and no other Extended Code is assigned |
| D0 | | Hardware Diagnostic | DeviceSelf Testing and Hardware Diagnostic Conditions |
| | 00 | | Reserved |
| | 01 | | Checksum (or CRC) error – Code space/ROM – Boot section |
| | 02 | | Checksum (or CRC) error – Code space/ROM – Application section |
| | 03 | | Checksum (or CRC) error – NV (flash/EEPROM) memory |
| | 04 | | Invalid non-volatile (NV) memory – Configuration bad |
| | 05 | | Invalid non-volatile (NV) memory – No configuration established |
| | 06 | | RAM memory bad – The RAM memory in the device was determined to be experiencing inoperative cells |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Identity Object, Class Code: $01_{Hex}$

| General Status Code (in hex) | 8-bit Associated Extended Status Codes | Status Name | Description of Status |
|---|---|---|---|
| | 07 | | ROM/Flash Memory bad |
| | 08 | | Flash/EEPROM (NV) Memory Bad |
| | 09 | | Interconnect wiring error / signal path problem |
| | 0A | | Power problem – Over current |
| | 0B | | Power problem – Over voltage |
| | 0C | | Power problem – Under voltage |
| | 0D | | Internal Sensor problem |
| | 0E | | System Clock Fault |
| | 0F | | Hardware configuration does not match NV configuration |
| | 10 | | Watchdog Disabled/Idle |
| | 11 | | Watchdog Timer Expired |
| | 12 | | Device over temperature |
| | 13 | | Ambient temperature outside of operating limits |
| | 14 – EF | (reserved) | |
| | F0 – FE | | Vendor specific Extended Codes |
| | FF | | Used with all General Codes when required and no other Extended Code is assigned |
| D1 | | Device Status/States | Device Status Events and Conditions |
| | 01 | | Power Applied |
| | 02 | | Device Reset |
| | 03 | | Device Power Loss |
| | 04 | | Activated |
| | 05 | | Deactivated |
| | 06 | | Enter Self-Test State |
| | 07 | | Enter Standby State |
| | 08 | | Enter Operational State |
| | 09 | | Non-Specific Minor Recoverable Fault Detected |
| | 0A | | Non-Specific Minor Unrecoverable Fault Detected |
| | 0B | | Non-Specific Major Recoverable Fault Detected |
| | 0C | | Non-Specific Major Unrecoverable Fault Detected |
| | 0D | | Fault(s) corrected |
| | 0E | | CCV Changed |
| | 0F | | Heartbeat Interval Changed |
| | 10 – EF | (reserved) | |
| | F0 - FE | Vendor Specific | Vendor Specific |
| | FF | | Used with all General Codes when required and no other Extended Code is assigned |
| D2 – EF | | Object Specific General Codes | Reserved by CIP – Not yet assigned |
| | 00 - FF | Reserved | |
| F0 – FF | | Vendor Specific General Codes | A vendor specific error has been encountered. The Additional Code Field of the Error Response defines the particular error encountered. Use of this General Error Code should only be performed when none of the Error Codes presented in this |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| General Status Code (in hex) | 8-bit Associated Extended Status Codes | Status Name | Description of Status |
|---|---|---|---|
| | | | table or within an Object Class definition accurately reflect the error. |
| | 00 - FF | Vendor Specific Extended Codes | All Extended Status Codes are available for association with each Vendor Specific General Code |

## 5A-2.6 Behavior

The behavior of the Identity Object is illustrated in the State Transition Diagram (STD) in Figure 5A-2.1. This STD associates the state of the device with the status reported by the Status Attribute with the state of the Module Status LED.

**Important:** A device may not be able to communicate in the Major Unrecoverable Fault state. Therefore, it might not be able to report a Major Unrecoverable Fault. It will not process a Reset service. The only exit from a Major Unrecoverable Fault is to cycle power.

The Identity object triggers production of heartbeat messages as defined by the underlying network when:

- the interval configured in the Heartbeat Interval Attribute has passed since the last heartbeat message.
- the Heartbeat message contents change, at a maximum rate of one “data changed” heartbeat message per second.

The Heartbeat Interval value shall be saved as a Non-Volatile attribute. Heartbeat messages are only triggered after the device has successfully completed the network access state machine and is online. Not all networks support sending the Heartbeat message.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5A-2.2 State Transition Diagram for Identity Object**

The STD for the Identity object contains the following events:

- **Power Applied** - the device is powered up
- **Passed Tests** - the device has successfully passed all self tests
- **Failed Tests** – The device's self test failed
- **Activated** - the device's configuration is valid and the application for which the device was designed is now capable of executing (communications channels may or may not yet be established)
- **Deactivated** - the device's configuration is no longer valid and the application for which the device was designed is no longer capable of executing (communication channels may or may not still be established)
- **Minor fault** - a fault classified as either a minor unrecoverable fault or a minor recoverable fault has occurred
- **Major recoverable fault** - an event classified as Major Recoverable Fault has occurred
- **Major unrecoverable fault** - an event classified as a Major Unrecoverable Fault has occurred
- **Fault Corrected** – the source of the Major Recoverable Fault has been corrected.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-2.13 State Event Matrix for Identity Object**

| Event | Nonexistent | Device Self Testing | Standby | Operational | Major Unrecoverable Fault | Major Recoverable Fault |
|---|---|---|---|---|---|---|
| Power Loss | Not Applicable | Transition to Nonexistent | Transition to Nonexistent | Transition to Nonexistent | Transition to Nonexistent | Transition to Nonexistent |
| Power Applied | Transition to Device Self Testing | Not Applicable | Not Applicable | Not Applicable | Not Applicable | Not Applicable |
| Failed Tests | Not Applicable | Transition to Major Unrecoverable Fault | Not Applicable | Not Applicable | Not Applicable | Not Applicable |
| Passed Tests | Not Applicable | Transition to Standby | Not Applicable | Not Applicable | Not Applicable | Not Applicable |
| Deactivated | Not Applicable | Ignore Event | Ignore Event | Transition to Standby | Ignore Event | Ignore Event |
| Activated | Not Applicable | Ignore Event | Transition to Operational | Ignore Event | Ignore Event | Ignore Event |
| Major Recoverable Fault | Not Applicable | Not Applicable | Transition to Major Recoverable Fault | Transition to Major Recoverable Fault | Ignore Event | Ignore Event |
| Major Unrecoverable Fault | Not Applicable | Not Applicable | Transition to Major Unrecoverable Fault | Transition to Major Unrecoverable Fault | Ignore Event | Ignore Event |
| Minor Recoverable Fault | Not Applicable | Ignore Event | Ignore Event | Ignore Event | Ignore Event | Ignore Event |
| Minor Unrecoverable Fault | Not Applicable | Ignore Event | Ignore Event | Ignore Event | Ignore Event | Ignore Event |
| Fault Corrected | Not Applicable | Not Applicable | Not Applicable | Not Applicable | Not Applicable | Transition to Standby |
| Reset | Not Applicable | Restart Self Tests | Transition to Device Self Testing | Transition to Device Self Testing | Ignore Event | Transition to Device Self Testing |
| Module Status LED | Off | Flashing Red/Green | Flashing Green | Solid Green | Solid Red | Flashing Red |

The SEM for the Identity object contains the following states:

- **Nonexistent** - the device is without power
- **Device Self Testing** - the device is executing its self tests
- **Standby** - the device needs commissioning due to an incorrect or incomplete configuration
- **Operational** - the device is operating in a fashion that is normal for the device
- **Major Recoverable Fault -** the device has experienced a fault that is believed to be recoverable
- **Major Unrecoverable Fault** - the device has experienced a fault that is believed to be unrecoverable


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-3 Message Router Object

## Class Code: 02 Hex

The Message Router Object provides a messaging connection point through which a Client may address a service to any object class or instance residing in the physical device.

## 5A-3.1 Class Attributes

**Table 5A-3.1 Message Router Object Class Attributes**

| Number | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |

## 5A-3.2 Instance Attributes

**Table 5A-3.2 Message Router Object Instance Attributes**

| Number | Need in implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Optional | Get | Object_list | STRUCT of | A list of supported objects | Structure with an array of object class codes supported by the device |
| | | | Number | UINT | Number of supported classes in the classes array | The number of class codes in the classes array |
| | | | Classes | ARRAY of UINT | List of supported class codes | The class codes supported by the device |
| 2 | Optional | Get | Number Available | UINT | Maximum number of connections supported | Count of the max number of connections supported |
| 3 | Optional | Get | Number active | UINT | Number of connections currently used by system components | Current count of the number of connections allocated to system communication |
| 4 | Optional | Get | Active Connections | ARRAY of: UINT | A list of the connection IDs of the currently active connections | Array of system connection IDs |

## 5A-3.3 Common Services

The Message Router Object provides the following Common Services:

**Table 5A-3.3 Message Router Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0Ehex | Conditional* | Conditional* | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 01hex | Optional | Optional | Get_Attributes_All | Returns the contents of all attributes |

*The Get_Attribute_Single service is REQUIRED if any attributes are implemented.

### 5A-3.3.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-3.4 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 |
| 4 | STRUCT of: | Optional Attribute List | |
| | UINT | number of attributes | 0 |
| | ARRAY of UINT | optional attributes | (null) |
| 5 | STRUCT of: | Optional Service List | |
| | UINT | number of services | 0 |
| | ARRAY of UINT | optional services | (null) |
| 6 | UINT | Maximum ID Number Class Attributes | 0 |
| 7 | UINT | Maximum ID Number Instance Attributes | 0 |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-3.5 Get_Attributes_All Response Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | STRUCT of: | Object_list | |
| | UINT | Number | 0 |
| | ARRAY of UINT | Classes | (null) |
| 2 | UINT | Number Available | 0 |
| 3 | UINT | Number Active | 0 |
| 4 | ARRAY of UINT | Active Connections | (null) |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

## 5A-3.4 Object–Specific Services

The Message Router Object provides the following Object-specific services.

**Table 5A-3.6 Object Specific Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| $4B_{hex}$ | Optional | n/a | SymbolicTranslation | Provides a translation from a Symbolic Segment path encoding to the equivalent Logical Segment path encoding, if it exists |

### 5A-3.4.1 Symbolic Translation

The SymbolicTranslation object specific service provides a mechanism for a device, that has implemented Symbolic Segment representations of internal path addresses, to translate those Symbolic Addresses encodings into their equivalent Logical Segment path encodings.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The service data of the SymbolicTranslation service shall be a single Symbolic Segment or ANSI Extended Symbol path address. The response data shall be the equivalent Logical path encoding.

**Table 5A-3.7 SymbolicTranslation Request Service Data Field Parameters**

| Name | Data Type | Description of Service Parameter |
|---|---|---|
| Symbolic Address | Padded EPATH | A Symbolic Segment or ANSI Extended Symbol Segment to be translated |

**Table 5A-3.8 SymbolicTranslation Response Service Data Field Parameters**

| Name | Data Type | Description of Service Parameter |
|---|---|---|
| Logical Address | Padded EPATH | A Logical Segment and optional Data Segment encoding representing the internal equivalent of the requested Symbolic Address |

**Table 5A-3.9 SymbolicTranslation Service Error Response Values**

| Error Code | Extended Code | Error Name | Description |
|---|---|---|---|
| | | | |
| 0x20 | 0x00 | Symbolic Path unknown | The Symbolic path is not recognized by the processing node |
| | 0x01 | Symbolic Path destination not assigned | The Symbolic path, although recognized, is not presently associated with a Logical path equivalent. |
| | 0x02 | Symbolic Path segment error | The symbol identifier or the symbol segment syntax was not understood by the processing node. |

## 5A-3.5 Behavior

The Message Router Object receives explicit messages and performs the following functions:

- processes the Electronic Key segment (if it exists)
- processes Network segments (if they exist)
- interprets the application path (e.g.: Class Instance) specified in a message
- routes a service to the specified object
- interprets services directed to it
- routes a response back to the originator of the request

### 5A-3.5.1 Service Request

The Eletronic Key segment is processed, if it exists. Table 5A-3.10 defines the Electronic Key segment error response values.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-3.10 Electronic Key Segment Error Response Values**

| General Status | Extended Status [1] | Status Name | Description |
|---|---|---|---|
| 0x25 | 0x114 | VENDOR ID OR PRODUCT CODE MISMATCH | The Product Code or Vendor ID specified in the electronic key logical segment does not match the Product Code or Vendor ID of the target device. If the compatibility bit is set this error code is returned when the requested Vendor ID or Product Code is 0 or the device cannot emulate the specified Vendor ID or Product Code. |
| | 0x115 | DEVICE TYPE MISMATCH | The Device Type specified in the electronic key logical segment does not match the Device Type of the target device. If the compatibility bit is set this error code is returned when the requested Device Type is 0 or the device cannot emulate the specified Device Type.<br>EXCEPTION: If the compatibility bit is set, Device Type 0 is allowed if the recipient can emulate a device that reports the deprecated Generic Device Type Number |
| | 0x116 | REVISION MISMATCH | The major and minor revision specified in the electronic key logical segment does not correspond to a valid revision of the target device. If the compatibility bit is set this error code is returned when the requested Major Revision and/or Minor Revision is 0 or the device cannot emulate the specified revision. |

Table Footnotes:

1. Note that these extended status values are the same used by the Connection Manager General Status 0x01.

Interpretation of the application path (e.g.: Class Instance) is performed on every service received by the Message Router.

If the application path can be interpreted, the service is then routed internally to a target object.

Any application path that cannot be interpreted by a device's implementation of a Message Router will report the Object_Not_Found error.

### 5A-3.5.2 Service Response

All service responses are routed back to the requesting device using the same path, in reverse.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-4 DeviceNet Object

## Class Code: 03 Hex

The DeviceNet Object provides the configuration and status of a DeviceNet port. Each DeviceNet product must support one (and only one) DeviceNet object per physical connection to the DeviceNet communication link.

See the Volume 3, DeviceNet Adaptation of CIP for the definition of this object class.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-5 Assembly Object

## Class Code: 04 Hex

The Assembly Object binds attributes of multiple objects, which allows data to or from each object to be sent or received over a single connection. Assembly objects can be used to bind input data or output data. The terms ”input” and ”output” are defined from the network’s point of view. An input will produce data on the network and an output will consume data from the network.

**Table 5A-5.1 Revision History**

| Assembly Class Revision | Description |
|---|---|
| 00 | Pre-release definition |
| 01 | Initial release |
| 02 | 1. Class specific Service Codes $4B_{hex}$ and $4C_{hex}$ obsoleted |

Assembly objects instances can either be dynamic or static:

- ***Dynamic:*** assemblies with member lists created and managed by the user. The member list can be altered by adding or deleting members. Dynamic assemblies shall be assigned instance Ids in the vendor specific range.
- ***Static:*** assemblies with member lists defined by the open device profile (see Chapter 6, Device Profiles) or vendor specific device profile. The Instance number, number of members, and member list are fixed.

**Important:** Instances of the Assembly Object are divided into the following address ranges to provide for extensions to device profiles.

**Table 5A-5.2 Assembly Instance ID Ranges**

| Range | | Open Device Profile Usage | Vendor Specific Device Profile Usage | Quantity |
|---|---|---|---|---|
| Hex | Decimal | | | |
| 0x01 - 0x63 | 1 - 99 | Open - Defined by the device profile | Vendor Specific | 99 |
| 0x64 - 0xC7 | 100 - 199 | Vendor Specific | Vendor Specific | 100 |
| 0xC8 - 0xD1 | 200 - 209 | Open - Defined by the device profile | Vendor Specific | 10 |
| 0xD2 - 0xEF | 210 - 239 | Reserved by CIP for future use | Reserved by CIP for future use | 30 |
| 0xF0 - 0xFF | 240 - 255 | Vendor Specific | Vendor Specific | 16 |
| 0x0100 - 0x02FF | 256 - 767 | Open - Defined by the device profile | Vendor Specific | 512 |
| 0x0300 - 0x04FF | 768 - 1279 | Vendor Specific | Vendor Specific | 512 |
| 0x0500 - 0xFFFF | 1280 - 65,535 | Open - Defined by the device profile | Vendor Specific | 64,256 |
| 0x00010000 - 0x000FFFFF | 65,536 - 1,048,575 | Open - Defined by the device profile | Vendor Specific | 983,040 |
| 0x00100000 - 0xFFFFFFFF | 1,048,576 - 4,294,967,295 | Reserved by CIP for future use | Reserved by CIP for future use | 4,293,918,720 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-5.1 Class Attributes

**Table 5A-5.3 Assembly Object Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Conditional* | Get | Revision | UINT | Revision of this object | The current value assigned to this attribute is two (02). |
| 2 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |
| *This attribute is REQUIRED if Instance Attribute 2 is supported, otherwise this attribute is OPTIONAL. | | | | | | |

## 5A-5.2 Instance Attributes

**Table 5A-5.4 Assembly Object Instance Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Conditional | Get | Number of Members in List | UINT | | Required for Dynamic Assembly only |
| 2 | Conditional | Set for Dynamic/ Get for Static | Member List | ARRAY of STRUCT: | The member list is an array of CIP paths | Required for Dynamic Assembly only |
| | | | Member Data Description | UINT | Size of member data. | Size in bits |
| | | | Member Path Size | UINT | Size of Member Path (in bytes). | |
| | | | Member Path | Packed EPATH | See Appendix C for the format of this field. | |
| 3 | Required | Set | Data | ARRAY of BYTE | | |
| 4 | Optional | Get | Size | UINT | Number of bytes in Attribute 3 | |

## 5A-5.3 Common Services

The Assembly Object provides the following Common Services:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-5.5 Assembly Object Common Services**

| Service Code | Need in Implementation | | | | Service Name | Description of Service |
|---|---|---|---|---|---|---|
| | Static Assembly | | Dynamic Assembly | | | |
| | Class | Instance | Class | Instance | | |
| 0Ehex | Conditional[1] | Required | Required | Required | Get_Attribute_ Single | Returns the contents of the specified attribute. |
| 08hex | n/a | n/a | Required | n/a | Create | Instantiates an Assembly Object within a specified class. Response contains instance number. Dynamic assemblies shall be assigned instance Ids in the vendor specific range. |
| 10hex | n/a | Optional | n/a | Conditional[2] | Set_Attribute_ Single | Modifies an attribute value. |
| 09hex | n/a | n/a | Optional[3] | Required | Delete | Deletes an Assembly Object and releases all associated resources. |
| 1Ahex | n/a | n/a | n/a | Conditional[4] | Insert_ Member | Adds a member to the Assembly Member List. |
| 1Bhex | n/a | n/a | n/a | Conditional[4] | Remove_ Member | Removes a member from the Assembly Member List. |
| 18hex | n/a | Optional | n/a | Optional | Get_Member | Returns a member from the Assembly Member List or Data attributes. |
| 19hex | n/a | Optional | n/a | Optional | Set_Member | Modifies a member of the Assembly Member List or Data attributes. |

Table Footnotes:

1 Required if Max Instance Class Attribute is implemented or Instance Attribute 2 (Member List).

2 If you choose NOT to support the Set_Attribute_Single common service at the Instance level for a Dynamic Assembly, then your product shall support the Insert_Member and Remove_Member common services.

3 At the class level this service deletes all existing Assembly instances.

4 If you choose NOT to support the Insert_Member and Remove_Member common services at the Instance level for a Dynamic Assembly, then your product shall support the Set_Attribute_Single common service.

See Appendix A for definition of these services

## 5A-5.4 Object–specific Services

The Assembly Object provides no Object–specific services. The following Object-specific service codes have been obsoleted:

**Table 5A-5.6 Assembly Object Obsolete Object-Specific Services**

| Obsoleted Service Code | Need in Implementation | | | | Service Name | Description of Service |
|---|---|---|---|---|---|---|
| | Static Assembly | | Dynamic Assembly | | | |
| | Class | Instance | Class | Instance | | |
| 4B hex | n/a | n/a | n/a | n/a | Add_Member | Obsolete |
| 4C hex | n/a | n/a | n/a | n/a | Remove_Member | Obsolete |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-5.5 Behavior

The behavior of the Assembly Object differs by the type of Assembly: *dynamic* or *static*. A Dynamic Assembly's member list is created and managed by the user of the device. The manager of the dynamic assembly must specify and maintain the member list.

A Static Assembly's member list is defined by the device manufacturer. It cannot be modified.

To provide for the ability to create and delete objects, as well as change member lists, Dynamic Assemblies support additional services which are not supported by Static Assemblies.

The following rules apply to the member lists of both static and dynamic assemblies.

When an empty path (Member Path Size = 0) is used in an assembly member list, the assembly inserts/discards the number of bits as specified in the Member Data Size field when producing/consuming. The assembly shall use a value of zero for all produced data which has been inserted. The use of an empty consumed path allows, for example, data destined for multiple nodes to be sent in a single message since each node can be configured to discard data in the message not intended for it.

The empty path shall be supported for all dynamic assemblies.

No checking is done by the Assembly Object at any time to verify that the size of the member data is correct for the given member path. It is the responsibility of the assembly member to properly handle too much or too little data. The assembly is required to deliver the configured number of bits to the member.

No padding is done by the Assembly Object to align data from each assembly member on a byte, word, or other boundary.

### 5A-5.5.1 Static Assemblies

The following State Transition Diagram, State Event Matrix and Attribute Access Table illustrate the behavior of Static assemblies.

**Figure 5A-5.1 Static Assembly State Transition Diagram**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-5.7 Static Assembly State Event Matrix**

| Event | Static Assembly Object State | | |
|---|---|---|---|
| | **Non-existent** | **Inactive** | **Active** |
| Power Up | Transition to Inactive | Not applicable | Not applicable |
| Get_Attribute _Single | Error: Object does not exist. (General Error Code $16_{hex}$) | Validate/service the request Return response | Validate/service the request Return response |
| Set_Attribute _Single | Error: Object does not exist. (General Error Code $16_{hex}$) | Validate/service the request Return response | Error: Object State Conflict (General Error Code $0C_{hex}$) |
| Message produced/ consumed | Error: Object does not exist. (General Error Code $16_{hex}$) | Begin producing/consuming from/to each member in list Transition to Active | Error: Object State Conflict (General Error Code $0C_{hex}$) |
| End of production/ consumption | Error: Object does not exist. (General Error Code $16_{hex}$) | Error: Object State Conflict (General Error Code $0C_{hex}$) | Transition to Inactive |

**Table 5A-5.8 Static Assembly Object Attribute Access**

| Attribute | Static Assembly Object State | | |
|---|---|---|---|
| | **Non-existent** | **Inactive** | **Active** |
| Number_of_Members | Not available | Read Only | Read Only |
| Member_List | Not available | Read Only | Read Only |
| Data | Not available | Read/Write | Read Only |

## 5A-5.5.2 Dynamic Assemblies

The State Transition Diagram, State Event Matrix and Attribute Access Table below illustrate the behavior of Dynamic assemblies.

**Figure 5A-5.2 Dynamic Assembly State Transition Diagram**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-5.9 Dynamic Assembly State Event Matrix**

| Event | Dynamic Assembly Object State | | |
|---|---|---|---|
| | **Non-Existent** | **Inactive** | **Active** |
| Create | Class instantiates an Assembly Object. Transition to Inactive | Not applicable | Not applicable |
| Delete | Error: Object does not exist. (General Error Code 0x16) | Release all associated resources. Transition to Non–Existent | Error: Object State Conflict (General Error Code 0x0C) |
| Get_Attribute_Single | Error: Object does not exist. (General Error Code 0x16) | Validate/service the request. Return response | Validate/service the request. Return response |
| Set_Attribute_Single | Error: Object does not exist. (General Error Code 0x16) | Validate/service the request. Return response | Error: Object State Conflict (General Error Code 0x0C) |
| Insert_Member | Error: Object does not exist. (General Error Code 0x16) | Validate/service the request. Return response | Error: Object State Conflict (General Error Code 0x0C) |
| Remove_Member | Error: Object does not exist. (General Error Code 0x16) | Validate/service the request. Return response | Error: Object State Conflict (General Error Code 0x0C) |
| Message produced/ consumed | Error: Object does not exist. (General Error Code 0x16) | Begin producing/consuming from/to each member in list. Transition to Active. | Error: Object State Conflict (General Error Code 0x0C) |
| End of production/ consumption | Error: Object does not exist. (General Error Code 0x16) | Error: Object State Conflict (General Error Code 0x0C) | Transition to Inactive |

**Table 5A-5.10 Dynamic Assembly Object Attribute Access**

| Attribute | Dynamic Assembly Object State | | |
|---|---|---|---|
| | **Non-existent** | **Inactive** | **Active** |
| Number_of_Members | Not available | Read Only | Read Only |
| Member_List[1] | Not available | Read/Write | Read Only |
| Data | Not available | Read/Write | Read Only |

Table Footnotes:

1 This attribute can be set by either the Insert_Member service (one member at a time) or the Set_Attribute_Single service (all members at once).

## 5A-5.5.3 Connection Points

Connection Points within the Assembly Object are identical to Instances. For example, Connection Point 4 of the Assembly Object is the same as Instance 4. Specifying a path of “20 04 24 VV 30 03” is the same as “20 04 2C VV 30 03”.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-6 Connection Object

## Class Code: 05 Hex

Use the Connection Object to manage the characteristics of a communication connection.

See Chapter 3 for the definition of this object class.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-7 Connection Manager Object

## Class Code: 06 Hex

Use this object for connection and connectionless communications, including establishing connections across multiple subnets.

See Chapter 3 for the definition of this object class.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-8 Register Object

## Class Code: 07 Hex

Use this object to address individual bits or a range of bits up to 64K bits of data.

Note that a Register object can operate as either an input register or an output register. The terms “input” and “output” are defined from the network’s point of view. An input will produce data on the network and an output will consume data from the network.

## 5A-8.1 Class Attributes

**Table 5A-8.1 Register Object Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |

## 5A-8.2 Instance Attributes

**Table 5A-8.2 Register Object Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Get | Bad Flag | BOOL | | 0=good<br>1=bad |
| 2 | Required | Set | Direction | BOOL | Direction of data transfer | 0=Input Register,<br>1=Output Register |
| 3 | Required | Set | Size | UINT | Size of register data in bits | |
| 4 | Required | Set (Set is optional if Direction=1) | Data | ARRAY of BITS | Data to be transferred | * |
| * All encoded bits are to be LSB aligned, as shown in this example. | | | | | | |

| | | | | |
|---|---|---|---|---|
| 7 . . . . . 0 | 15 .. . . . . 8 | 23 .. . . . . 16 | 29 . . . . . 24 | 30 bit array encoding example |

## 5A-8.3 Common Services

The Register Object provides the following Common Services:

**Table 5A-8.3 Register Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0Ehex | Conditional* | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 10hex | n/a | Optional | Set_Attribute_Single | Modifies an attribute value. |
| *The Get_Attribute_Single service is REQUIRED if any attributes are implemented. | | | | |
| See Appendix A for definition of these services | | | | |

## 5A-8.4 Object–specific Services

The Register Object provides no Object–specific services.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-8.5 Behavior

The State Transition Diagram, State Event Matrix, and Attribute Access Table below illustrate the behavior of the Register Object.

**Figure 5A-8.1 Register Object State Transition Diagram**

**Table 5A-8.4 Register Object State Event Matrix**

| Event | State | |
|---|---|---|
| | **Non-existent** | **Run** |
| Power up | Transition to Run | Not applicable. |
| Get_Attribute_Single | Error: Object does not exist (General Error Code $16_{hex}$) | Validate/service the request. Return response. |
| Set_Attribute_Single | Error: Object does not exist (General Error Code $16_{hex}$) | Validate/service the request. Return response. |
| Message produced/consumed | Error: Object does not exist (General Error Code $16_{hex}$) | Apply/retrieve data. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-9 Discrete Input Point Object

## Class Code: 08 Hex

The Discrete Input Point (DIP) Object models discrete inputs in a product. You can use this object in applications as simple as a toggle switch or as complex as a discrete I/O control module. Note that the term ”input” is defined from the network’s point of view. An input will produce data on the network.

The Discrete Input Point interface is to real input points such as a switch or screw terminal. The input is sampled and the data is stored in this object’s VALUE attribute. A sample of the discrete input value is triggered via an external command (input change-of-state, cyclic data trigger, etc.)

## 5A-9.1 Revision History

Since the initial release of this object class definition changes have been made that require a revision update of this object class. The table below represents the revision history:

**Table 5A-9.1 Discrete Input Point Object Revision History**

| Revision | Reason for object definition update |
|---|---|
| 01 | Initial Definition at First Release of Specification |
| 02 | IDLE state removed from this object’s behavior |

## 5A-9.2 Class Attributes

**Table 5A-9.2 Discrete Input Point Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Get | Revision | UINT | Revision of this object | The current value assigned to this attribute is two (02). |
| 2 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-9.3 Instance Attributes

**Table 5A-9.3 Discrete Input Point Object Instance Attributes**

| Attribute ID | Need in Implem. | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Optional | Get | | Number of Attributes | USINT | Number supported in this product | |
| 2 | Optional | Get | | Attribute List | ARRAY OF USINT | List of attributes supported in this product | |
| 3 | Required | Get | | Value | BOOL | Input point value | 0=off; 1=on |
| 4 | Optional | Get | | Status | BOOL | Input point status | 0=OK; 1=product specific alarm or status |
| 5 | Optional | Set | | Off_On Delay | UINT | filter time for off to on transition 0 - 65,535 microseconds [1] | The default value is 0. |
| 6 | Optional | Set | | On_Off Delay | UINT | filter time for on to off transition 0 - 65,535 microseconds [2] | The default value is 0. |
| 7 | Optional | Set[3] | NV | Off_On Cycles | UDINT | Total number of times the *Value* attribute transitioned from the Off to the On state | [default] = 0 |

Table Footnotes:

1 The input must be on for the amount of filter time specified by the OFF_ON DELAY attribute before the ON state is recorded in the VALUE attribute.

2 The input must be off for the amount of filter time specified by the ON_OFF DELAY attribute before the OFF state is recorded in the VALUE attribute.

3 Applications including this attribute may restrict its access to Get Only; thus, allowing the application support of a factory initialized cycle life indication.

## 5A-9.4 Common Services

The Discrete Input Point Object provides the following Common Services:

**Table 5A-9.4 Discrete Input Point Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0Ehex | Required | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 01hex | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this objects attributes (See the Get_Attributes_All Response definition below) |
| 10hex | n/a | Optional | Set_Attribute_Single | Modifies an attribute value. |
| 02hex | Optional | Optional | Set_Attributes_All | Modifies the value of a list of attributes (See the Set_Attributes_All Request definition below) |
| See Appendix A for definitions of these common services. | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-9.4.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-9.5 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 [1] |
| 2 | UINT | Max Instance | 0 |
| 6 | UINT | Maximum ID Number Class Attributes | 1 |
| 7 | UINT | Maximum ID Number Instance Attributes | 3 |

Table Footnotes:

1. The default value only applies for Revision 1 of this object. Newer revisions require that this attribute be implemented, which will override this default value.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-9.6 Get_Attributes_All Response Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | USINT | Number of Attributes | 0 |
| 2 | ARRAY of USINT | Attribute List | (null) |
| 3 | BOOL[1] | Value | |
| 4 | BOOL[1] | Status | 0 |
| 5 | UINT | Off_On Delay | 0 |
| 6 | UINT | On_Off Delay | 0 |

Table Footnotes:

1. The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

### 5A-9.4.2 Set_Attributes_All Request

No settable attributes currently exist at the Class level for the Discrete Input Point Object.

At the **Instance level**, the order of attributes passed in the Set_Attributes_All request is as follows:

**Table 5A-9.7 Set_Attributes_All Request Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 5 | UINT | Off_On_Delay | |
| 6 | UINT | On_Off_Delay | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Important:** The Set_Attributes_All service is to be supported only if **all** settable attributes shown above are implemented as settable.

## 5A-9.5 Object–specific Services

The DIP Object provides no Object–specific services.

## 5A-9.6 Behavior

The State Transition Diagram in Figure 5A-9.1 provides a graphical description of the events and corresponding state transitions. A subset of the states and events may be supported in an application, but the behavior must be consistent.

The State Event Matrix in Table 5A-9.9 lists all pertinent events and the corresponding action to be taken while in each state.

**Important:** Events can occur simultaneously, but the *Fault* events have priority if they occur simultaneously with other events.

The following behavior is related to the *Off_On Cycles* attribute. The value of *Off_On Cycles* is incremented once for each time the input transitions from an 'Off' state to a 'On' state. The purpose is to assist in tracking the cycles the device that is reporting the input has encountered. Upon reaching a terminal count (maximum value) and experiencing another cycle to count, the *Off_On Cycles* value will "wrap" to a value of zero (0) and continue to count cycles as described.

**Figure 5A-9.1 State Transition Diagram for Discrete Input Point Object**

**Important:** Events can occur simultaneously, but the *Fault* events have priority if they occur simultaneously with other events.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The following SEM contains these states:

- ***Non–Existent:*** a module without power.
- ***Available:*** waiting for a connection, power–up discrete input point defaults are set.
- ***Run:*** DIP sensing data from its input and transmitting the data.
- ***Recoverable Fault:*** a recoverable fault has occurred.
- ***Unrecoverable Fault:*** an unrecoverable fault has occurred.

The SEM also contains these events:

**Table 5A-9.8 Discrete Input Point Object Event Definitions**

| This event | Is |
|---|---|
| Sample Trigger | a change of state, cyclic timer trigger, application trigger |
| Connection Deleted | I/O connection deleted. |
| Apply_Attributes | the Apply service of the I/O connection object the Discrete Input Point object is connected to. Note: the application is responsible for validating the connection object's attributes. |
| Fault Cleared | the application clearing a detected fault |
| Connection Transitions to Established | I/O connection transitions to Established. |
| Connection Transitions to Timed Out state | I/O connection transitions to Timed-Out |

The figure below is a conceptual illustration of the state machine for a typical input object (producing application). The events listed above are represented by the dotted line labeled "state change."

**Figure 5A-9.2 State Machine Illustration for Typical Discrete Input Point Object**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-9.9 State Event Matrix for the Discrete Input Point Object**

| Event | State | | | | |
|---|---|---|---|---|---|
| | Non-Existent | Available | Run | Recoverable Fault | Unrecoverable Fault |
| Sample Trigger | Not Applicable | Ignore event | Sample data, Send data | Ignore event | Ignore event |
| Apply Attributes | Not Applicable | Verify attributes, return result | Return error (Object State Conflict) | Return error (Object State Conflict) | Ignore event |
| Connection Deleted | Not Applicable | Ignore Event | Transition to Available | Transition to Available | Ignore event |
| Connection Transitions to Established | Not Applicable | Transition to Run | Ignore event | Ignore event | Ignore event |
| Connection Transitions to Timed Out state | Not Applicable | Ignore event | Transition to Recoverable Fault | Ignore event | Ignore event |
| Fault Cleared | Not Applicable | Not Applicable | Not Applicable | Transition to Run | Ignore event |
| Get_Attribute | Return Error (Object Does Not Exist) | Return value | Return value | Return value | Ignore event |
| Set_Attribute | Return Error (Object Does Not Exist) | Accept value | Accept value | Accept value | Ignore event |
| I/O Status LED | Off | Off | Solid Green | Flash Red | Solid Red |

### 5A-9.6.1 Attribute Access Rules

Except in the Non-Existent and Unrecoverable Fault states, all attributes are gettable or settable according to their access rules.

Because the only required Instance attribute is *Value*, the only required behavior of a Discrete Input Point is that it indicates a boolean value of OFF or ON. The optional attributes either provide more information about the discrete input point or alter the behavior of the input point.

**Figure 5A-9.3 Discrete Input Point Object Value Attribute Illustration**

#### 5A-9.6.1.1 Status Instance Attribute

The optional *Status* Instance attribute is simply a logical OR of all possible failure or alarm conditions for the point.

**Figure 5A-9.4 Discrete Input Point Object Status Attribute Function**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-10 Discrete Output Point Object

## Class Code: 09 Hex

A Discrete Output Point (DOP) models discrete outputs in a product. You can use this object in applications such as discrete I/O control modules, relays, switches, etc. Note that the term "output" is defined from the network's point of view. An output will consume data from the network.

The Discrete Output Point interface is to real output points such as a relay or LED. The output is read from this object's VALUE attribute and applied to the output terminal (e.g. screw terminal).

## 5A-10.1 Class Attributes

**Table 5A-10.1 Discrete Output Point Object Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |

## 5A-10.2 Instance Attributes

**Table 5A-10.2 Discrete Output Point Object Instance Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Optional | Get | Number of Attributes | USINT | Number of attributes supported in this product | |
| 2 | Optional | Get | Attribute List | ARRAY OF USINT | List of attributes supported in this product | |
| 3 | Required | Set | Value | BOOL | Output point value | 0=off; 1=on |
| 4 | Optional | Get | Status | BOOL | Output point status | 0=OK; 1=failure or alarm |
| 5 | Optional | Set | Fault Action | BOOL | Action taken on output's value in Recoverable Fault state | 0=Fault Value attribute; 1=hold last state |
| 6 | Optional | Set | Fault Value | BOOL | User–defined value for use with Fault Action attribute | 0=off; 1=on |
| 7 | Optional | Set | Idle Action | BOOL | Action taken on output's value in Idle state | 0=Idle Value attribute; 1=hold last state |
| 8 | Optional | Set | Idle Value | BOOL | User-defined value for use with Idle Action attribute | 0=off; 1=on |
| 9 | Optional | Set | Run_Idle_ Command | BOOL | Generates the Receive_Idle or Receive_ Ready_to_ Run event | 0=Receive_Idle 1=Receive_Ready_to _ Run |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 10 | Optional | Set | Flash | BOOL | Flash output at periodic rate if point is ON | 0=no flash;<br>1=flash |
| 11 | Optional | Set | Flash Rate | USINT | Flash Rate for Flash attribute | unsigned positive integer indicating frequency in Hz, e.g. 1= 1Hz |
| 12 | Optional | Get | Object State | USINT | State of the object | 1 = Non-Existent<br>2 = Available<br>3 = Idle<br>4 = Ready<br>5 = Run<br>6 = Recoverable Fault<br>7 = Unrecoverable Fault<br>255 = Reserved |
| 13 | Conditional [1] | | Test Output Mode for Safety | See CIP Safety Specification (Volume 5, Chapter 5) | | |

Table Footnotes:

1 This attribute is not allowed if the device is not a safety device. If the device is a safety device, see Volume 5.

**Important:** Optional attributes either provide more information about the discrete output point or alter the behavior of the output point. If the following optional instance attributes are not supported the default attribute values below indicate the required behavior for instances of this object class:

**Table 5A-10.3 Optional Attribute Default Values**

| Attribute ID | Name | Default Value |
|---|---|---|
| 5 | Fault Action | 0 |
| 6 | Fault Value | 0 |
| 7 | Idle Action | 0 |
| 8 | Idle Value | 0 |
| 10 | Flash | 0 |

### 5A-10.2.1 Value

The only required Instance attribute is *Value*. The required behavior of a Discrete Output Point's value is that it outputs a boolean value of OFF or ON.

**Figure 5A-10.1 Discrete Output Point Object Value Attribute Illustration**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-10.2.2 Status

The *Status* attribute is simply a logical OR of all possible failure or alarm conditions for the point.

**Figure 5A-10.2 Discrete Output Point Status Attribute**

### 5A-10.2.3 Fault and Idle Attributes

These optional attributes define a safe state for the DOP when in the Idle or Recoverable Fault states:

- Fault Action
- Fault Value
- Idle Action
- Idle Value

**Table 5A-10.4 Fault/Idle Action Pair Definition**

| This attribute pair | Defines |
|---|---|
| Fault Action and Fault Value | the value of the DOP in the Fault state. |
| Idle Action and Idle Value | the value of the DOP in the Idle state. |

The following dependencies exist among these attributes:

**Table 5A-10.5 Fault/Idle Action Pair Dependencies**

| If this attribute is supported | Then this attribute must also be supported by definition |
|---|---|
| Fault Action | Fault Value (although this attribute could be defined to always be OFF). |
| Idle Action | Idle Value (although this attribute could be defined to always be OFF). |

The “Action” attributes dictate what the DOP will do upon entering that state. The DOP will either hold its value in the last state or update it to the value stored in the corresponding Fault or Idle Value attribute.

Upon entering the Recoverable_Fault state the DOP will behave according to the following table.

**Table 5A-10.6 Recoverable Fault State Behavior**

| | Fault Action = 0 | Fault Action = 1 |
|---|---|---|
| Fault Value = 0 | DOP uses the value in the Fault_Value attribute (0) to update its value. | DOP leaves Value in last state. Fault_Value attribute has no effect. |
| Fault Value = 1 | DOP uses the value in the Fault_Value attribute (1) to update its value. | DOP leaves Value in last state. Fault_Value attribute has no effect. |

The Idle attributes follow similar behavior.

**Important:** There is one deviation from the behavior specified in the table above. If the DOP enters the Recoverable_Fault state from the Idle state in response to the I/O connection transitioning to Timed Out, the DOP’s value should go unchanged. This is shown in the DOP’s State Transition Diagram.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-10.2.4 Run_Idle Command

The *Run_Idle Command* attribute causes the Receive_Ready_to_Run or Receive_Idle event to be sent to the DOP. Refer to the DOP's State Transition Diagram to see the resulting transitions. This attribute only has effect when the Discrete Output object is in the Idle, Ready, or Run states. While in the Available or Recoverable Fault state, an attempt to set this attribute will result in an Object_State_Conflict error. A read (Get_Attribute_Single) of this attribute will result in a zero being returned always.

### 5A-10.2.5 Flash

The *Flash* attribute modifies the behavior of the DOP such that when the DOP is in the ON state, the output flashes OFF and ON at a periodic rate.

### 5A-10.2.6 Flash Rate

The *Flash Rate* attribute modifies the behavior of the DOP in that when the DOP is in the ON state and the Flash attribute is ON, it sets the periodic flash rate by an unsigned positive integer.

## 5A-10.3 Common Services

The Discrete Output Object provides the following Common Services:

**Table 5A-10.7 Discrete Output Point Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0E hex | Conditional* | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 10 hex | n/a | Required | Set_Attribute_Single | Modifies an attribute value. |
| 01 hex | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this objects attributes (See the Get_Attributes_All Response definition below) |
| 02 hex | n/a | Optional | Set_Attributes_All | Modifies the contents of the attributes of the class or object. |
| *The Get_Attribute_Single service is REQUIRED if any attributes are implemented. | | | | |
| See the Appendix A for definitions of these common services. | | | | |

### 5A-10.3.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-10.8 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 |
| 2 | UINT | Max Instance | 0 |
| 6 | UINT | Maximum ID Number Class Attributes | 0 |
| 7 | UINT | Maximum ID Number Instance Attributes | 3 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-10.9 Get_Attributes_All Response Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | USINT | Number of Attributes | 0 |
| 2 | ARRAY of USINT | Attribute List | (null) |
| 3 | BOOL[1] | Value | |
| 4 | BOOL[1] | Status | 0 |
| 5 | BOOL[1] | Fault Action | 0 |
| 6 | BOOL[1] | Fault Value | 0 |
| 7 | BOOL[1] | Idle Action | 0 |
| 8 | BOOL[1] | Idle Value | 0 |
| 10 | BOOL[1] | Flash | 0 |
| 11 | USINT | Flash Rate | 0 |
| 12 | USINT | Object State | 255 |

Table Footnotes:

1. The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

### 5A-10.3.2 Set_Attributes_All Request

No settable attributes currently exist at the **Class level** for the Discrete Output Point object.

At the **Instance level**, the order of attributes passed in the Set_Attributes_All request is as follows:

**Table 5A-10.10 Set_Attributes_All Request Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 3 | BOOL[1] | Value | |
| 5 | BOOL[1] | Fault Action | |
| 6 | BOOL[1] | Fault Value | |
| 7 | BOOL[1] | Idle Action | |
| 8 | BOOL[1] | Idle Value | |
| 10 | BOOL[1] | Flash | |
| 11 | USINT | Flash Rate | |

Table Footnotes

1. The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Important:** The Set_Attributes_All service is to be supported only if **all** settable attributes shown above are implemented as settable.

## 5A-10.4 Object–specific Services

The Discrete Output Point Object provides no Object–specific services.

## 5A-10.5 Behavior

The State Transition Diagram in Figure 5A-10.3 provides a graphical description of the events and corresponding state transitions. A subset of the states and events may be supported in an application, but the behavior must be consistent.

The State Event Matrix in this section lists all pertinent events and the corresponding action to be taken while in each state.

**Important:** Events can occur simultaneously, but the *Fault* events have priority if they occur simultaneously with other events. In addition, if the Receive_Data event occurs simultaneously with any other event, the other event takes precedence.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5A-10.3 State Transition Diagram for Discrete Output Point Object**

The SEM contains the following states:

- ***Non–Existent:*** module without power.
- ***Available:*** DOP defaults configured, waiting for connection.
- ***Idle:*** DOP in Idle mode and does not apply received data.
- ***Ready:*** waiting for valid data to apply.
- ***Run:*** DOP applying received data to its output.
- ***Recoverable Fault:*** a recoverable fault has occurred.
- ***Unrecoverable Fault:*** an unrecoverable fault has occurred.

The SEM also contains these events:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-10.11 Discrete Output Point Event Definitions**

| This event | Is |
|---|---|
| Receive_Data | an event that signals the reception of I/O data and causes the object to transition to the Run state. |
| Receive_Fault | an event that is internally generated and product-specific. The network does not know if a Fault has occurred. |
| Receive_Idle | the setting of the Run_Idle Command attribute to the value 0 -or- the IO connection object receives an I/O message ***containing no application data*** |
| Receive _Run | the setting of the Run_Idle Command attribute to the value 1. |
| Apply_Attributes | the Apply service of the I/O connection object the Discrete Output Point object is connected to. Note: the application is responsible for validating the connection object's attributes. |
| Connection Deleted | I/O connection deleted. |
| Connection Transitions to Established | I/O connection transitions to Established. |
| Connection transitions to the Timed Out state | the expiration of the connection timer. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-10.12 State Event Matrix for the Discrete Output Point Object**

| Event | State | | | | | | |
|---|---|---|---|---|---|---|---|
| | Non - Existent | Available | Idle | Ready | Run | Recoverable Fault | Unrecoverable Fault |
| Receive_Data | Not applicable | Not applicable | If data length is non-zero transition to Run<br>Verify and Accept Data<br>Transition to Run Ignore event | LED Solid Green, Accept Data, Transition to Run | If data length is zero [2]<br>Transition to Idle<br>Otherwise<br>Verify and Accept Data | Ignore event | Ignore event |
| Receive_Fault | Not applicable | Not applicable | Transition to Recoverable Fault [1] | Transition to Recoverable Fault [1] | Transition to Recoverable Fault [1] | Ignore event | Ignore event |
| Receive_Idle | Not applicable | Return Error (object state conflict) | Ignore event | Transition to Idle [2] | Transition to Idle [2] | Return Error (object state conflict) | Ignore event |
| Receive_Ready_ to_Run | Not applicable | Return Error (object state conflict) | Transition to Ready | Ignore event | Ignore event | Return Error (object state conflict) | Ignore event |
| Validate_Connection _Attributes | Not applicable | Verify attributes, return results | Return Error (object state conflict) | Return Error (object state conflict) | Return Error (object state conflict) | Return Error (object state conflict) | Ignore event |
| Connection Deleted | Not applicable | Ignore event | Transition to Available | Transition to Available | Transition to Available | Transition to Available | Ignore event |
| Connection Transition to Established | Not applicable | Transition to Ready | Return Error (object state conflict) | Return Error (object state conflict) | Return Error (object state conflict) | Transition to Ready [3] | Ignore event |
| Connection transitions to Timed Out state | Not applicable | Not applicable | Transition to Recoverable Fault | Transition to Recoverable Fault | Transition to Recoverable Fault [1] | Ignore event | Ignore event |
| Get_Attribute | Return Error (Object Does Not Exist) | Return value | Return value | Return value | Return value | Return value | Ignore event |
| Set_Attribute | Return Error (Object Does Not Exist) | Verify, and Accept, and Apply value | Verify, and Accept, and Apply value | Verify, and Accept, and Apply value | Verify, and Accept, and Apply value | Verify, and Accept, and Apply value | Ignore event |
| Fault_Cleared | Ignore event | Ignore event | Ignore event | Ignore event | Ignore event | Transition to Ready [3] | Ignore event |

Table Footnotes:

1. Fault: Hold Last State OR use Fault Value.
2. Idle: Hold Last State OR use Idle Value.
3. If no other faults exist (Note, a Connection time out is considered a fault)

### 5A-10.5.1 Attribute Access

Except in the Non-Existent and Unrecoverable Fault states, all attributes are gettable or settable according to their access rules.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-11 Analog Input Point Object

## Class Code: 0A Hex

The Analog Input Point (AIP) Object models analog inputs in a product. It can be used in applications as simple as a single analog point and as complex as an analog I/O control module. Note that the term ”input” is defined from the network’s point of view. An input will produce data on the network.

The Analog Input Point interface is to real input points such as a thermocouple or pressure transducer. The input is sampled and the data is stored in this object’s VALUE attribute. A sample of the analog input value is triggered via an external command (input change-of-state, cyclic data trigger, etc.)

## 5A-11.1 Revision History

Since the initial release of this object class definition changes have been made that require a revision update of this object class. The table below represents the revision history:

**Table 5A-11.1 Analog Input Point Object Revision History**

| Revision | Reason for object definition update |
|---|---|
| 01 | Initial Definition at First Release of Specification |
| 02 | IDLE state removed from this object’s behavior |

## 5A-11.2 Class Attributes

**Table 5A-11.2 Analog Input Point Object Class Attributes**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Get | Revision | UINT | Revision of this object | The current value assigned to this attribute is two (02). |
| 2 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-11.3 Instance Attributes

**Table 5A-11.3 Analog Input Point Object Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Optional | Get | Number of Attributes | USINT | Number of attributes supported | |
| 2 | Optional | Get | Attributes List | Array of USINT | List of attributes supported by the point | |
| 3 | Required | Get | Value | INT or based on attribute 8 | Analog input value. The data type defaults to INT but may be changed based on attribute 8. | |
| 4 | Optional | Get | Status | BOOL | Indicates if a fault or alarm has occurred. | 0= operating without alarms or faults.<br>1=alarm or fault condition exists, the Value attribute may not represent the actual field value. |
| 5 | Optional | Get | Owner Vendor ID | UINT | Vendor ID of channel's owner | |
| 6 | Optional | Get | Owner Serial Number | UDINT | 32–bit serial number of channel's owner | |
| 7 | Optional | Set | Input Range | USINT | Input range the point is operating in | 0 = -10V to 10V<br>1 = 0V to 5V<br>2 = 0V to 10V<br>3 = 4mA to 20mA<br>4 = -15mV to 75mV<br>5 = -15mV to 30mV<br>6 = -5V to 5V<br>7 = 1V to 5V<br>8 = 0mA to 20mA<br>9 = 0mA to 50mA<br>10 – 99 = Reserved<br>100-131=Vendor Specific<br>132–154 =Reserved<br>255 = Only return with Get_Attributes_All if attribute not supported |
| 8 | Optional | Set | Value Data Type | USINT | Determines the data type of Value | 0 = INT<br>1 = REAL<br>2 = USINT<br>3 = SINT<br>4 = DINT<br>5 = LINT<br>6 = UINT<br>7 = UDINT<br>8 = ULINT<br>9 = LREAL<br>100 = vendor specific |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-11.4 Common Services

The Analog Input Object provides the following Common Services:

**Table 5A-11.4 Analog Input Point Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0Ehex | Required | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 10hex | n/a | Conditional[1] | Set_Attribute_Single | Modifies an attribute value. |
| 01hex | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this objects attributes (See the Get_Attributes_All Response definition below) |
| 02hex | n/a | Optional | Set_Attributes_All | Modifies the contents of the attributes of the class or object. |

Table Footnotes:
1 The Set_Attribute_Single service is required only if Instance Attributes 7 and/or 8 are implemented.

See Appendix A for definition of these services

### 5A-11.4.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-11.5 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 [1] |
| 2 | UINT | Max Instance | 0 |
| 6 | UINT | Maximum ID Number Class Attributes | 0 |
| 7 | UINT | Maximum ID Number Instance Attributes | 3 |

Table Footnotes:
1. The default value only applies for Revision 1 of this object. Newer revisions require that this attribute be implemented, which will override this default value.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-11.6 Get_Attributes_All Response Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | USINT | Number of Attributes | 0 |
| 2 | ARRAY of USINT | Attribute List | (null) |
| 8 | USINT | Value Data Type | 0 |
| 3 | Specified by Value Data Type (Attribute #8) | Value | |
| 4 | BOOL[1] | Status | 0 |
| 5 | UINT | Owner Vendor ID | 0 |
| 6 | UDINT | Owner Serial Number | 0 |
| 7 | USINT | Input Range | 255[2] |

Table Footnotes:

1. The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.
2. The default value of 255 shall be returned for this attribute ONLY if it is not supported by the device.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

### 5A-11.4.2 Set_Attributes_All Request

No settable attributes currently exist at the **Class level** for the Analog Input Point object.

At the **Instance level**, the order of attributes passed in the Set_Attributes_All request is as follows:

**Table 5A-11.7 Set_Attributes_All Request Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 7 | USINT | Input Range | |
| 8 | USINT | Value Data Type | |

**Important:** The Set_Attributes_All service is to be supported only if **all** settable attributes shown above are implemented as settable.

## 5A-11.5 Object–specific Services

The Analog Input Object provides no Object–specific services:

## 5A-11.6 Behavior

The State Transition Diagram in Figure 5A-11.1 provides a graphical description of the events and corresponding state transitions. A subset of the states and events may be supported in an application, but the behavior must be consistent.

The State Event Matrix in this section lists all pertinent events and the corresponding action to be taken while in each state. A subset of the states and events may be supported in an application, but the behavior must still be consistent.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5A-11.1 State Transition Diagram for Analog Input Point Object**

1. Indicated by the application LED = I/O Status LED

**Important:** Events can occur simultaneously, but the *Fault* events have priority if they occur simultaneously with other events.

The SEM contains the following states:

- ***Non–Existent:*** module with no power.
- ***Available:*** waiting for a connection, power–up analog input point defaults are set.
- ***Run:*** AIP sensing data from its input and transmitting the data.
- ***Recoverable Fault:*** a recoverable fault has occurred.
- ***Unrecoverable Fault:*** an unrecoverable fault has occurred.

The SEM also contains these events:

**Table 5A-11.8 Analog Input Point Object Event Definitions**

| This event | Is |
|---|---|
| Sample Trigger | a change of state; cyclic timer trigger; non–zero length Bit Strobe or Poll Command. |
| Connection Deleted | I/O connection deleted. |
| Apply_Attributes | the Apply service of the I/O connection object the Analog Input Point object is connected to. Note: the application is responsible for validating the connection object's attributes. |
| Fault Cleared | the application clearing a detected fault |
| Connection Transitions to Established | I/O connection transitions to Established. |
| Connection Transitions to Timed Out state | I/O connection transitions to Timed-Out |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Figure 5A-11.2 shows a conceptual illustration of the state machine for a typical input object (producing application). The events listed above are represented by the dotted line labeled "state change."

**Figure 5A-11.2 State Machine Illustration for Typical Analog Input Point Object**

**Table 5A-11.9 State Event Matrix for the Analog Input Point Object**

| Event | State | | | | |
|---|---|---|---|---|---|
| | **Non-Existent** | **Available** | **Run** | **Recoverable Fault** | **Unrecoverable Fault** |
| Sample Trigger | Not Applicable | Ignore event | Sample data, Send data | Ignore event | Ignore event |
| Apply Attributes | Not Applicable | Verify attributes, return result | Return error (Object State Conflict) | Return error (Object State Conflict) | Ignore event |
| Connection Deleted | Not Applicable | Ignore Event | Transition to Available | Transition to Available | Ignore event |
| Connection Transitions to Established | Not Applicable | Transition to Run | Ignore event | Ignore event | Ignore event |
| Connection Transitions to Timed Out state | Not Applicable | Ignore event | Transition to Recoverable Fault | Ignore event | Ignore event |
| Fault Cleared | Not Applicable | Not Applicable | Not Applicable | Transition to Run | Ignore event |
| Get_Attribute | Return Error (Object Does Not Exist) | Return value | Return value | Return value | Ignore event |
| Set_Attribute | Return Error (Object Does Not Exist) | Accept value | Accept value | Accept value | Ignore event |
| I/O Status LED | Off | Flash Green | Solid Green | Flash Red | Solid Red |

### 5A-11.6.1 Attribute Access Rules

Except in the Non-Existent and Unrecoverable Fault states, all attributes are gettable or settable according to their access rules.

Because the only required Instance attribute is *Value*, the only required behavior of an AIP is that it indicates an analog value.

Optional attributes either provide more information about the AIP or alter the behavior of the input point.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-11.6.1.1 Status Attribute

The optional *Status* attribute is simply a logical OR of all possible failure or alarm conditions for the point.

**Figure 5A-11.3 Analog Input Point Object Status Attribute Operation**

### 5A-11.6.1.2 Owner Vendor ID, Owner Serial Number

These attributes may be implemented in both the point and group objects. If implemented in both, the values shall be the same for the point objects and the group object that binds the point object. If a connection to this object is made using the Connection Manager, then the value for the Owner Vendor Id and Owner Serial Number shall be the Originator Vendor Id and Originator Serial Number contained in the Forward_Open request. These values shall be cleared when the connection is closed or deleted.

### 5A-11.6.1.3 Input Range

The *Input Range* attribute determines the set of analog values within which the inputs may operate. The value of the Input Range affects the default and legal values that other attributes may have.

Input Range is necessary only if the point may be switched to operate within different ranges, which changes the behavior of the point. For example, if the point is configured to operate from 0V to 5V or from 4mA to 20mA, the variability of ranges will likely change the expected behavior of the Value attribute and perhaps change vendor specific attributes.

### 5A-11.6.1.4 Value Data Type

The *Value Data Type* attribute determines the data type to be used by the attribute Value (and perhaps other attributes as well). If Value Data Type is not used, then Value defaults to INT. Value (and other attributes) may behave as REAL, USINT, or any other data type based upon the definition of Value Data Type, which ultimately provides the ability for the analog point to be defined in any length necessary.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-12 Analog Output Point Object

## Class Code: 0B Hex

The Analog Output Point (AOP) models the point level attributes and services of the analog outputs in a product. It can be used to model voltage output or parts of an analog I/O control module. Note that the term ”output” is defined from the network’s point of view. An output will consume data from the network. The Analog Output Point interface is to real output points such as a Motor Operated Valve (MOV) or Linear Positioner. The output is read from this object’s VALUE attribute and applied to the output terminal (e.g. screw terminal).

## 5A-12.1 Class Attributes

**Table 5A-12.1 Analog Output Point Object Class Attributes**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Get | Revision | UINT | Revision of this object | The current value assigned to this attribute is two (02). |
| 2 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |

## 5A-12.2 Instance Attributes

**Table 5A-12.2 Analog Output Point Object Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Optional | Get | Number of Attributes | USINT | Number of attributes supported | 0–255 |
| 2 | Optional | Get | Attribute List | Array of USINT | List of attributes supported by the point | |
| 3 | Required | Set | Value | INT or based on attribute 8 | Analog output value. The data type defaults to INT but may be changed based on attribute 8. | |
| 4 | Optional | Get | Status | BOOL | Indicates if a fault or alarm has occurred. | 0= operating without alarms or faults.<br>1=alarm or fault condition exists, the Value attribute may not represent the actual field value. |
| 5 | Optional | Get | Owner - Vendor ID | UINT | Vendor ID of channel’s owner | |
| 6 | Optional | Get | Owner - Serial Number | UDINT | 32–bit serial number of channel’s owner | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Analog Output Point Object, Class Code: $0B_{Hex}$

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 7 | Optional | Set | Output Range | USINT | Specifies the output range the output channel is to use | 0 = 4mA to 20mA<br>1 = 0V to 10V<br>2 = 0mA to 20mA<br>3 = -10V to 10V<br>4 = 0V to 5V<br>5 = -5V to 5V<br>6 = 1V to 5V<br>7 - 8 = Reserved<br>9 = 0mA to 50mA<br>10 – 99 = Reserved<br>100-131=Vendor Specific<br>132 – 254 = Reserved<br>255 = Only return with Get_Attributes_All if attribute not supported |
| 8 | Optional | Set | Value Data Type | USINT | Determines the data type of Value | 0 = INT (default)<br>1 = REAL<br>2 = USINT<br>3 = SINT<br>4 = DINT<br>5 = LINT<br>6 = UINT<br>7 = UDINT<br>8 = ULINT<br>9 = LREAL<br>100 = vendor specific |
| 9 | Optional | Set | Fault Action | USINT | Output value to go to on failure or fault | 0 = hold last state (default)<br>1 = low limit[1]<br>2 = high limit<br>3 = use Fault Value<br>4 - 99 = reserved<br>100 - 199 = vendor specific<br>200 - 255 = reserved |
| 10 | Optional | Set | Idle Action | USINT | Output value to go to on idle mode | 0 = hold last state (default)<br>1 = low limit[1]<br>2 = high limit<br>3 = use Idle Value<br>4 - 99 = reserved<br>100 - 199 = vendor specific<br>200 - 255 = reserved |
| 11 | Required if Fault Action is implemented | Set | Fault Value | INT or based on attribute 8 | User defined value outputs go to in fault mode if Fault Action = 3, user specified value | |
| 12 | Required if Idle Action is impelmented | Set | Idle Value | INT or based on attribute 8 | User defined value outputs go to in idle mode if Idle Action = 3, user specified value | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 13 | Required if Idle Action is implemented | Set | Command | BOOL | Changes state of AOP to Idle Mode or Run Mode | 0 = idle<br>1 = run |
| 14 | Optional | Get | Object State | USINT | State of the object | 1 = Non-Existent<br>2 = Available<br>3 = Idle<br>4 = Ready<br>5 = Run<br>6 = Recoverable Fault<br>7 = Unrecoverable Fault<br>255 = Reserved |

Table Footnotes:
1 All values 0, 1, 2, and 3 shall be supported when the Fault Action or Idle Action attributes are implemented

**Important:** If the following optional instance attributes are not supported, the default attribute values below indicate the required behavior for instances of this object class:

**Table 5A-12.3 Analog Output Point Default Values for Unsupported Attributes**

| Attribute ID | Name | Default Value |
|---|---|---|
| 7 | Output Range | 0 |
| 8 | Value Data Type | 0 |
| 9 | Fault Action | 0 |
| 10 | Idle Action | 0 |
| 11 | Fault Value | minimum value of range (attribute 7) |
| 12 | Idle Value | 0 |

## 5A-12.3 Common Services

The Analog Output Object provides the following Common Services:

**Table 5A-12.4 Analog Output Point Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0E hex | Conditional* | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 10 hex | n/a | Required | Set_Attribute_Single | Modifies an attribute value. |
| 01 hex | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this objects attributes (See the Get_Attributes_All Response definition below) |
| 02 hex | n/a | Optional | Set_Attributes_All | Modifies the contents of the attributes of the class or object. |

*The Get_Attribute_Single service is REQUIRED if any class attributes are implemented.

See the Appendix A for definitions of these common services.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-12.3.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-12.5 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 [1] |
| 2 | UINT | Max Instance | 0 |
| 6 | UINT | Maximum ID Number Class Attributes | 0 |
| 7 | UINT | Maximum ID Number Instance Attributes | 3 |

Table Footnotes:

1. The default value only applies for Revision 1 of this object. Newer revisions require that this attribute be implemented, which will override this default value.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-12.6 Get_Attributes_All Response Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 8 | USINT | Value Data Type | 0 |
| 3 | Specified by Value Data Type (Attribute #8) | Value | |
| 1 | USINT | Number of Attributes | 0 |
| 2 | ARRAY of USINT | Attribute List | (null) |
| 4 | BOOL[1] | Status | 0 |
| 5 | UINT | Owner Vendor ID | 0 |
| 6 | UDINT | Owner Serial Number | 0 |
| 7 | USINT | Output Range | 255[2] |
| 9 | USINT | Fault Action | 0 |
| 10 | USINT | Idle Action | 0 |
| 11 | Specified by Value Data Type (Attribute #8) | Fault Value | Minimum value of Output Range (Attribute #7) |
| 12 | Specified by Value Data Type (Attribute #8) | Idle Value | 0 |
| 13 | BOOL[1] | Command | 0 |
| 14 | USINT | Object State | 255[2] |

Table Footnotes:

1. The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.
2. The default value of 255 shall be returned for this attribute ONLY if it is not supported by the device.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-12.3.2 Set_Attributes_All Request

No settable attributes currently exist at the **Class level** for the Analog Output Point object.

At the **Instance level**, the order of attributes passed in the Set_Attributes_All request is as follows:

**Table 5A-12.7 Set_Attributes_All Request Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 8 | USINT | Value Data Type | |
| 3 | Specified by Value Data Type (Attribute #8) | Value | |
| 7 | USINT | Output Range | |
| 9 | USINT | Fault Action | |
| 10 | USINT | Idle Action | |
| 11 | Specified by Value Data Type (Attribute #8) | Fault Value | |
| 12 | Specified by Value Data Type (Attribute #8) | Idle Value | |
| 13 | BOOL[1] | Command | |

Table Footnotes:

1. The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.

**Important:** The Set_Attributes_All service is to be supported only if **all** settable attributes shown above are implemented as settable.

## 5A-12.4 Object–specific Services

The Analog Output Point Object provides no Object–specific services.

## 5A-12.5 Behavior

The State Transition Diagram in Figure 5A-12.1 provides a graphical description of the events and corresponding state transitions. A subset of the states and events may be supported in an application, but the behavior must be consistent.

The State Event Matrix in Table 5A-12.9 lists all pertinent events and the corresponding action to be taken while in each state. A subset of the states and events may be supported in an application, but the behavior must still be consistent.

**Important:** Events can occur simultaneously, but the *Fault* events have priority if they occur simultaneously with other events. In addition, if the Receive_Data event occurs simultaneously with any other event, the other event takes precedence.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5A-12.1 State Transition Diagram for Analog Output Point Object**

The following SEM contains these states:

- ***Non–Existent:*** module without power.
- ***Available:*** AOP defaults configured, waiting for connection.
- ***Idle:*** AOP in standby mode and does not apply received data.
- ***Ready*** (to run): waiting for valid data to apply.
- ***Run:*** AOP applying received data to its output.
- ***Recoverable Fault:*** a recoverable fault has occurred.
- ***Unrecoverable Fault:*** an unrecoverable fault has occurred.

The SEM also contains these events:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-12.8 Analog Output Point Object Event Definitions**

| This event | Is |
|---|---|
| Receive_Data | an event that signals the reception of I/O data and causes the object to transition to the Run state. |
| Receive_Fault | an event that is internally generated and product-specific. The network does not know if a Fault has occurred. |
| Receive_Idle | the setting of the COMMAND attribute to the value 0 -or- within the Master/Slave paradigm the event signaled when the I/O connection object receives a Bit–Strobe or Poll Command message ***containing no application data*** |
| Receive_Run | the setting of the COMMAND attribute to the value 1 -or- within the Master/Slave paradigm the event signaled when the I/O connection object receives a Bit–Strobe or Poll Command message ***containing application data*** |
| Apply_Attributes | the Apply service of the I/O connection object the Analog Output Point object is connected to. Note: the application is responsible for validating the connection object's attributes. |
| Connection Deleted | I/O connection deleted. |
| Connection Transitions to Established | I/O connection transitions to Established. |
| Connection transitions to the Timed Out state | the expiration of the connection timer. |

The figure below is a conceptual illustration of the state machine for a typical output object (consuming application). The events listed above are represented by the dotted line labeled "state change."


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5A-12.2 State Machine Illustration for Typical Analog Output Point Object**

**Table 5A-12.9 State Event Matrix for the Analog Output Point Object**

| Event | State | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Non-Existent** | **Available** | **Idle** | **Ready** | **Run** | **Recoverable Fault** | **Unrecoverable Fault** |
| Receive_Data | Not applicable | Not applicable | Ignore event | LED Solid Green, Accept Data, Transition to Run | Accept Data | Ignore event | Ignore event |
| Receive_Fault | Not applicable | Not applicable | Transition to Recoverable Fault [1] | Transition to Recoverable Fault [1] | Transition to Recoverable Fault [1] | Ignore event | Ignore event |
| Receive_Idle | Not applicable | Not applicable | Ignore event | Transition to Idle [2] | Transition to Idle [2] | Transition to Idle [2] | Ignore event |
| Receive_Run | Not applicable | Not applicable | Transition to Ready | Ignore event | Ignore event | Transition to Ready | Ignore event |
| Apply_Attributes | Not applicable | Verify attributes, return results | Return Error (object state conflict) | Return Error (object state conflict) | Return Error (object state conflict) | Return Error (object state conflict) | Ignore event |
| Connection Deleted | Not applicable | Ignore event | Transition to Available | Transition to Available | Transition to Available | Transition to Available | Ignore event |
| Connection Transition to Established | Not applicable | Transition to Ready | Return Error (object state conflict) | Return Error (object state conflict) | Return Error (object state conflict) | Transition to Ready | Ignore event |
| Connection Transitions to Timed Out state | Not applicable | Not applicable | Transition to Recoverable Fault | Transition to Recoverable Fault | Transition to Recoverable Fault [1] | Ignore event | Ignore event |
| Get_Attribute | Return Error (Object Does Not Exist) | Return value | Return value | Return value | Return value | Return value | Ignore event |
| Set_Attribute | Return Error (Object Does Not Exist) | Accept value | Accept value | Accept value | Accept value | Accept value | Ignore event |

Table Footnotes:

1 Fault: Hold Last State OR use Fault Value.

2 Idle: Hold Last State OR use Idle Value.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-12.5.1 Attribute Access Rules

Except in the Non-Existent and Unrecoverable Fault states, all attributes are gettable or settable according to their access rules.

Because the only required Instance attribute is Value, the only required behavior of a Analog Output Point is that it outputs an analog value within the defined range for value (see Instance Attribute 8).

Optional attributes either provide more information about the AOP or alter the behavior of the output point.

#### 5A-12.5.1.1 Status Attribute - Optional

The optional Status attribute is simply a logical OR of all possible failure or alarm conditions for the point.

**Figure 5A-12.3 Analog Output Point Object Status Attribute Operation**

#### 5A-12.5.1.2 Command Attribute – Optional

The Command attribute explicitly controls whether the AOP is in the Run or Idle state. It has effect only when the Analog Output object is in the Idle, Ready, Recoverable Fault or Run states. While in the Available state, an attempt to set this attribute will result in an Object_State_Conflict error. A read (Get_Attribute_Single) of this attribute will result in a zero being returned always.

#### 5A-12.5.1.3 Fault and Idle Attributes – Optional

These optional attributes define a “safe state” for the analog output point when in the Idle or Fault states:

- Fault Action
- Fault Value
- Idle Action
- Idle Value

**Table 5A-12.10 Fault/Idle Action Pair Definition**

| This attribute pair | Defines |
|---|---|
| Fault Action and Fault Value | the value of the AOP when in the Recoverable Fault state |
| Idle Action and Idle Value | the value of the AOP when in the Idle state |

The following dependencies exist among these attributes:

**Table 5A-12.11 Fault/Idle Pair Dependencies**

| If this attribute is supported | Then this attribute must also be supported by definition |
|---|---|
| Fault Action | Fault Value (although this attribute could be defined to always be LOW). |
| Idle Action | Idle Value |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5A-12.5.1.4 Owner VendorId, Owner Serial Number

These attributes may be implemented in both the point and group objects. If implemented in both, the values shall be the same for the point objects and the group object that binds the point object. If a connection to this object is made using the Connection Manager, then the value for the Owner Vendor Id and Owner Serial Number shall be the Originator Vendor Id and Originator Serial Number contained in the Forward_Open request. These values shall be cleared when the connection is closed or deleted

#### 5A-12.5.1.5 Output Range Attribute – Optional

The Output Range attribute determines the set of analog values within which the outputs may operate. The value of the Output Range affects the default and legal values that other attributes may have.

Output Range is necessary only if the point may be switched to operate within different ranges, which changes the behavior of the point. For example, if the point is configured to operate from -10V to 10V or from 0V to 10V, the variability of ranges will likely affect the low value that outputs go to in safety state (you can use -10 V or 0 V, Output Range).

#### 5A-12.5.1.6 Value Data Type Attribute – Optional

The Value Data Type attribute determines the data type to be used by the attribute Value (and perhaps other attributes as well). If Value Data Type is not used, then Value defaults to INT. Value (and other attributes) may behave as REAL, USINT, or any other data type based upon the definition of Value Data Type, which ultimately provides the ability for the analog point to be defined in any length necessary.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-13 Presence Sensing Object

## Class Code: 0E Hex

This object senses the presence or absence of a real world target.

## 5A-13.1 Class Attributes

**Table 5A-13.1 Presence Sensing Object Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |

## 5A-13.2 Instance Attributes

**Table 5A-13.2 Presence Sensing Object Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Get | Output | BOOL | 0 = switching element open<br>1 = switching element closed | See Semantics" section |
| 2 | Optional | Get | Number of Attributes | USINT | Number of attributes supported | |
| 3 | Optional | Get | Attribute List | Array of USINT | List of attributes supported | |
| 4 | Optional | Get | Diagnostic | BOOL | 0=good<br>1=fault | |
| 5 | Optional | Set | On Delay | UINT | 0 – 65,535 ms | |
| 6 | Optional | Set | Off Delay | UINT | 0 – 65,535 ms | |
| 7 | Optional | Set | One Shot Delay | UINT | 0 – 65,535 ms | |
| 8 | Optional | Set | Operate Mode | BOOL | 0 = output attribute as specified<br>1 = inversion of output attribute | See Semantics" section |
| 9 | Optional | Set | Sensitivity | USINT | 0 – 255 | See Semantics" section |
| 10 | Optional | Get | Target Margin | USINT | 1 – 255 | See Semantics" section |
| 11 | Optional | Get | Background Margin | USINT | 0 – 100 | See Semantics" section |
| 12 | Optional | Set | Min Detect Distance | UINT | 0 – 65,535 mm | See Semantics" section |
| 13 | Optional | Set | Max Detect Distance | UINT | 0 – 65,535 mm * | See Semantics" section |
| 14 | Optional | Get | Detect Distance | UINT | 0 – 65,535 mm | |

* The Max Detect Distance must be greater than or equal to the Min Detect Distance


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-13.2.1 Semantics

#### 5A-13.2.1.1 Output

The Output value is determined by the switching element in the device. The state of this value is controlled by the:

- operation mode: *Light Operate (LO)/Dark Operate (DO)*
  *Normally Closed (NC)/Normally Open (NO)*
- detection of an object (absence or presence)
- timing
- type of device (photoelectric, proximity)

The following table indicates output values for common presence sensing devices.

**Table 5A-13.3 Common Output Values for Presence Sensing Devices**

| Presence Sensing Device | Output Values | |
|---|---|---|
| | **Output = On** | **Output = Off** |
| Diffuse Photoelectric,<br>Retroreflective Photoelectric | light detected (LO)<br>no light detected (DO) | no light detected (LO)<br>light detected (DO) |
| Through–Beam Photoelectric | light beam detected (LO)<br>light beam interrupted (DO) | light beam interrupted (LO)<br>light beam detected (DO) |
| Inductive Proximity, Capacitive,<br>Ultrasonic, Limit Switch | object present (NO)<br>object absent (NC) | object absent (NO)<br>object present (NC) |

#### 5A-13.2.1.2 Operate Mode

Use the Operate (Sensing) Mode to invert the level definition of the Output attribute.

**Table 5A-13.4 Common Output Values for Presence Sensing Devices**

| For | use the Operate Mode to: |
|---|---|
| photoelectric sensors, | set Light Operate (LO)/Dark Operate (DO) mode. |
| proximity sensors, | set Normally Open (NO)/Normally Closed (NC) mode. |

#### 5A-13.2.1.3 Sensitivity

Sensitivity is the amplification of the signal received by the detection device. Sensitivity is defined with either of two methods (i.e. low, medium, high or a numeric scale). This attribute accommodates both methods by converting the terms into a numeric scale (i.e. 0 = lowest, ... , n = highest).

#### 5A-13.2.1.4 Target Margin

Target Margin is the signal strength received by the device when the signal strength is above the device output switching threshold. The margin value is a scale of 1 to 255 with 1 representing the device output switching threshold.

#### 5A-13.2.1.5 Background Margin

Background Margin is the signal strength received by the device when the signal strength is below the device output switching threshold. The margin value is a scale of 0 to 100 where the value 100 equals the device switching threshold and the value 0 represents no signal received.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5A-13.2.1.6 Minimum and Maximum Detect Distances

The Minimum and Maximum Detect Distances define the range in which an object can be detected. If an object passes through the sensing device's field of view outside this range, the sensing device does not change the output (attribute number 1) of the device to indicate detection of that object.

## 5A-13.3 Common Services

The Presence Sensing Object provides the following Common Services:

**Table 5A-13.5 Presence Sensing Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0Ehex | Conditional* | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 10hex | n/a | Optional | Set_Attribute_Single | Modifies an attribute value. |
| *The Get_Attribute_Single service is REQUIRED if any attributes are implemented. | | | | |
| See the Appendix A for definitions of these common services. | | | | |

## 5A-13.4 Object–specific Services

The Presence Sensing Object provides no Object–specific services.

## 5A-13.5 Behavior

The behavior of the Presence Sensing Object is illustrated in Table 5A-13.1 Presence Sensing Object Class Attributes and Table 5A-13.6 Presence Sensing Object State Event Matrix. Table 5A-13.7 indicates the accessibility of attributes.

**Figure 5A-13.1 Presence Sensing Object State Transition Diagram**

**Table 5A-13.6 Presence Sensing Object State Event Matrix**

| Event | State | |
|---|---|---|
| | Non-Existent | Run |
| Connection established | Transition to Run | Ignore event |
| Connection deleted | Ignore event | Transition to Non-Existent |
| Get_Attribute_Single | Ignore event | Return value |
| Set_Attribute_Single | Ignore event | Accept value |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-13.7 Presence Sensing Object Attribute Access**

| Attribute | State | |
|---|---|---|
| | Non-Existent | Running |
| Output | Not available | Read Only |
| Number of Attributes | Not available | Read Only |
| Attribute List | Not available | Read Only |
| Diagnostic | Not available | Read Only |
| On Delay | Not available | Read/Write |
| Off Delay | Not available | Read/Write |
| One Shot Delay | Not available | Read/Write |
| Operate Mode | Not available | Read/Write |
| Sensitivity | Not available | Read/Write |
| Target Margin | Not available | Read Only |
| Background Margin | Not available | Read Only |
| Min Detect Distance | Not available | Read/Write |
| Max Detect Distance | Not available | Read/Write |
| Detect Distance | Not available | Read Only |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-14 Parameter Object

## Class Code: 0F Hex

Use of the Parameter Object provides a known, public interface to a device's configuration data. In addition, this object also provides all the information necessary to define and describe each of a device's individual configuration parameters.

This object allows a device to fully identify a configurable parameter by supplying a full description of the parameter, including minimum and maximum values and a human–readable text string describing the parameter.

There must be one instance of this object class for each of the device's configurable parameters. The instances must start at instance one and increment by one with no gaps in the instances.

Each instance is linked to one of the configurable parameters, which is typically (but is not required to be) an attribute of one of the device's other objects. Changing the *Parameter Value* attribute of a Parameter Object causes a corresponding change in the attribute value indicated by the *Link Path* attribute (the attribute value being pointed to by the Parameter Object).

For example, Instance #1 of the Parameter Object below identifies the parameter represented by the Analog Input Point (AIP) Object, Instance #1, Attribute #7.

**Figure 5A-14.1 Parameter Object representation of an Analog Input Point Attribute**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-14.1 Class Attributes

**Table 5A-14.1 Parameter Object Class Attributes**

| Attrib ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | This class attribute is optional and is described in Chapter 4 of this specification. | | | | | |
| 2 | Required | Get | Max Instance | UINT | Maximum instance number of an object currently created in this class level of the device. | The largest instance number of a created object at this class hierarchy level. |
| 3 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |
| 8 | Required | Get | Parameter Class Descriptor | WORD | Bits that describe parameters. | See Table 5A-14.2 |
| 9 | Required | Get | Configuration Assembly Instance | UINT | Instance number of the configuration assembly. | This attribute shall be set to zero if a configuration assembly is not supported. |
| 10 | Optional [1] | Set | Native Language | USINT | Language ID for all character array accesses. | 0=English<br>1=French<br>2=Spanish<br>3=Italian<br>4=German<br>5=Japanese<br>6=Portugese<br>7=Mandarin Chinese |

Table Footnotes:

1 If the Active Language attribute of the Identity Object (attribute ID 11) is supported, then this attribute shall not be supported.

### 5A-14.1.1 Semantics

#### 5A-14.1.1.1 Parameter Class Descriptor

The Parameter Class Descriptor attribute contains bits to describe parameter characteristics. The following bits are supported:

**Table 5A-14.2 Parameter Class Descriptor Bit Values**

| Bit | Definition |
|---|---|
| 0 | Supports Parameter Instances |
| 1 | Supports Full Attributes |
| 2 | Must do non-volatile storage save command |
| 3 | Params are stored in Non-Volatile storage |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5A-14.1.1.1.1 Supports Parameter Instances, Bit 0

*"Supports Parameter Instances"* indicates that Parameter Instances are present for each parameter. Parameter Instances contain all attributes or just the stub attributes. Bit values are:

**Table 5A-14.3 "Supports Parameter Instance" Bit Values**

| Value | Meaning |
|---|---|
| 1 | Individual Parameter instances ARE supported. |
| 0 | NO Parameter Instances are supported. Only configuration assembly instance used. |

#### 5A-14.1.1.1.2 Supports Full Attributes, Bit 1

*"Supports Full Attributes"* indicates that each Parameter Instance contains all of the attributes and are not just stubs. Bit values are:

**Table 5A-14.4 "Supports Full Attributes" Bit Values**

| Value | Meaning |
|---|---|
| 1 | All Full Parameter Attributes ARE supported. |
| 0 | Only Parameter Instance stub attributes are supported. |

#### 5A-14.1.1.1.3 Must Do NV Storage Save Command, Bit 2

*"Must Do NV Storage Save Command"* indicates that parameters are not stored into NV storage until a Save service is performed on the Parameter Object.

**Table 5A-14.5 "Must Do VN Storage Save Command "Bit Values**

| Value | Meaning |
|---|---|
| 1 | Must execute non–volatile storage save command. The *Save* service of the parameter class is used to save instance parameter values. |
| 0 | Do not have to execute non–volatile storage save command. |

#### 5A-14.1.1.1.4 Params Are Stored In Non-Volatile Storage, Bit 3

*"Params are stored in Non–Volatile Storage"* indicates that parameters are stored in non–volatile storage. Bit values are:

**Table 5A-14.6 "Params are stored in Non-Volatile Storage" Bit Values**

| Value | Meaning |
|---|---|
| 1 | All Full parameters are stored in non–volatile storage. |
| 0 | Parameters are not stored in non–volatile storage. |

A Parameter Object Stub is a shorthand version of a parameter object. The stub stores only the configuration data value, providing only a standard access point for the parameter.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-14.2 Instance Attributes

**Table 5A-14.7 Parameter Object Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Stub/ Full | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|---|
| 1 | Required | Get Set [2] | V | Stub | Parameter Value | Specified by Data Type Attribute (5). | Actual value of parameter. It can be read from or written to. Scaling shall not be performed to this attribute by the Parameter Object. Scaling, if enabled, shall be handled by the configuration tool. | |
| 2 | Required | Set | NV | Stub | Link Path Size | USINT | Size of link path. If this attribute is 0, then no link is specified. | Number of bytes. |
| 3 | Required | Set | NV | Stub | Link Path | Packed EPATH | CIP path to the object from where this parameter's value is retrieved. | The Link Path is limited to 255 bytes. See Appendix C. |
| 4 | Required | Get | NV | Stub | Descriptor | WORD | Description of parameter. | See Table 5A-14.8 |
| 5 | Required | Get | NV | Stub | Data Type | USINT | Data type code. | See Appendix C-6.1 |
| 6 | Required | Get | NV | Stub | Data Size | USINT | Number of bytes in Parameter Value | |
| 7 | Conditional [1] | Get | NV | Full | Parameter Name String | SHORT_ STRING | A human-readable string representing the parameter name. For example, "Frequency #1" | The maximum number of characters is 16 |
| 8 | Conditional [1] | Get | NV | Full | Units String | SHORT_ STRING | Engineering Unit String | The maximum number of characters is 4 |
| 9 | Conditional [1] | Get | NV | Full | Help String | SHORT_ STRING | Help String | The maximum number of characters is 64 |
| 10 | Conditional [3] | Get | NV | Full | Minimum Value | Specified by Data Type Attribute (5). | Generally, the minimum value to which the parameter can be set. | See semantics section 5A-14.2.1.3 |
| 11 | Conditional [3] | Get | NV | Full | Maximum Value | Specified by Data Type Attribute (5). | Generally, the maximum value to which the parameter can be set. | See semantics section 5A-14.2.1.3 |
| 12 | Conditional [1] | Get | NV | Full | Default Value | Specified by Data Type Attribute (5). | The actual value the parameter should be set to when the user wants the default for the parameter. | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Parameter Object, Class Code: $0F_{Hex}$

| Attr ID | Need in Implementation | Access Rule | NV | Stub/ Full | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|---|
| 13 | Conditional [1] | Get | NV | Full | Scaling Multiplier | UINT | Multiplier for Scaling Factor. | See Table 5A-14.12 |
| 14 | Conditional [1] | Get | NV | Full | Scaling Divisor | UINT | Divisor for Scaling Formula. | See Table 5A-14.12 |
| 15 | Conditional [1] | Get | NV | Full | Scaling Base | UINT | Base for Scaling Formula. | See Table 5A-14.12 |
| 16 | Conditional [1] | Get | NV | Full | Scaling Offset | INT | Offset for Scaling Formula. | See Table 5A-14.12 |
| 17 | Conditional [1] | Get | NV | Full | Multiplier Link | UINT | Parameter Instance of Multiplier source. | See Table 5A-14.13 |
| 18 | Conditional [1] | Get | NV | Full | Divisor Link | UINT | Parameter Instance of Divisor source. | See Table 5A-14.13 |
| 19 | Conditional [1] | Get | NV | Full | Base Link | UINT | Parameter Instance of Base source. | See Table 5A-14.13 |
| 20 | Conditional [1] | Get | NV | Full | Offset Link | UINT | Parameter Instance of Offset source. | See Table 5A-14.13 |
| 21 | Conditional [1] | Get | NV | Full | Decimal Precision | USINT | Specifies number of decimal places to use when displaying the scaled engineering value. Also used to determine actual increment value so that incrementing a value causes a change in scaled engineering value to this precision. | |
| 22 | Optional [4] | Get | NV | Full | International Parameter Name | STRINGI | Human readable string representing the parameter name | |
| 23 | Optional [4] | Get | NV | Full | International Engineering Units | STRINGI | Engineering units associated with the parameter | |
| 24 | Optional [4] | Get | NV | Full | International Help String | STRINGI | Help String | |

Table Footnotes:

1. Required if the Parameter Class Descriptor (class attribute 8) bit 1 is set (supports full attributes). Not allowed if the Parameter Object class descriptor bit 1 is not set.
2. Set is required if the Read Only Parameter bit (4) in the Descriptor attribute (4) is FALSE.
3. See Table 5A-14.11 if the Parameter Class Descriptor (class attribute 8) bit 1 is set (supports full attributes). Not allowed if the Parameter Class Descriptor (class attribute 8) bit 1 is not set.
4. Optional if the Parameter Class Descriptor (class attribute 8) bit 1 is set (supports full attributes). Not allowed if the Parameter Object class descriptor bit 1 is not set.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-14.2.1 Semantics

#### 5A-14.2.1.1 Descriptor – Attribute 4

The Descriptor Instance Attribute contains these bits, which describe the parameter:

**Table 5A-14.8 Semantics of Descriptor Instance Attribute**

| Bit | Definition | Meaning |
|---|---|---|
| 0 | Supports Settable Path | Indicates that link path can be set. |
| 1 | Get_Enumerated_String Supported | Indicates that enumerated strings can be read with the Get_Enum_String service. If descriptor bit 8 is set, enumerated values outside the min/max range may exist. |
| 2 | Supports Scaling | See Table 5A-14.9, Descriptor Scaling Bits Usage. |
| 3 | Supports Scaling Links | See Table 5A-14.9, Descriptor Scaling Bits Usage. |
| 4 | Read Only Parameter | Indicates that the Parameter Value attribute can only be read, and not set. |
| 5 | Monitor Parameter | Indicates that the Parameter Value attribute is updated in real time by the device. |
| 6 | Supports Extended Precision Scaling | See Table 5A-14.9, Descriptor Scaling Bits Usage. |
| 7 | Supports non-consecutive enumerated strings | Obsolete. |
| 8 | Allows both enumeration and min/max range values | Valid for integer data types only. Enumerated values and values in the min/max range are valid. Enumerated values may be inside or outside of the min/max range. If an enumerated value is inside the range, the enumerated string takes precedence over the value. |
| 9 | Non-displayed parameter | The configuration tool shall not display this parameter |
| 10 | Indirect Parameter Reference | The parameter value represents the instance number of another parameter. The referenced parameter instance shall not be an additional indirect reference. A parameter value of 0 shall indicate no reference. The data type for this parameter shall be USINT, UINT or UDINT. |
| 11 | Not Addressable | The parameter is not directly addressable from the network by any of the Set_Attribute or Get_Attribute services. |
| 12 | Save Supported | This bit, when set, indicates that this parameter may be individually saved by sending the Save service to this parameter instance. |
| 13 | Apply Supported | This bit, when set, indicates the execution of the Apply serice to this parameter instance is required before attribute changes affect instance behavior. |
| 14 | Write Only Parameter | Indicates that the Parameter Value attribute can only be set, and not read. |
| 15 | Reserved | This shall be set to zero |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-14.9 Descriptor Scaling Bits Usage**

| Supports Extended Precision Scaling (Bit 6) | Supports Scaling (Bit 2) | Supports Scaling Links (Bit 3) | Meaning |
|---|---|---|---|
| 0 | 0 | n/a | No Scaling |
| 0 | 1 | 0 | Supports scaling using Equation 5A-14.1 and Equation 5A-14.2. The scaling attributes (13-16) values are used. |
| 0 | 1 | 1 | Supports scaling using Equation 5A-14.1 and Equation 5A-14.2. For non-zero link attributes (17-20), the indicated parameter value will be used. Otherwise the corresponding scaling attribute (13-16) value is used. |
| 1 | n/a | 0 | Supports extended scaling using Equation 5A-14.3 and Equation 5A-14.4. The scaling attribute (13-16) values are used. |
| 1 | n/a | 1 | Supports extended scaling using Equation 5A-14.3 and Equation 5A-14.4. For non-zero link attributes (17-20), the indicatd parameter value will be used. Otherwise the corresponding scaling attribute (13-16) value is used. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-14.2.1.2 Data Type – Attribute 5

The Data Type Instance Attribute specifies the data type of the parameter. This value shall be specified according to the format shown in Appendix C, Section C-6.1. The following data types are obsolete but may be encountered in older devices:

**Table 5A-14.10 Obsolete Data Types Possible in the Parameter Object**

| Attr Value | Definition | Description |
|---|---|---|
| 1 | WORD | 16-bit word |
| 2 | UINT | 16-bit unsigned integer |
| 3 | INT | 16-bit signed integer |
| 4 | BOOL | Boolean |
| 5 | SINT | Short Integer |
| 6 | DINT | Double Integer |
| 7 | LINT | Long Integer |
| 8 | USINT | Unsigned Short Integer |
| 9 | UDINT | Unsigned Double Integer |
| 10 | ULINT | Unsigned Long Integer |
| 11 | REAL | Single floating point format (IEEE 754) |
| 12 | LREAL | Double floating point format (IEEE 754) |
| 13 | ITIME | Duration (short) |
| 14 | TIME | Duration |
| 15 | FTIME | Duration (high resolution) |
| 16 | LTIME | Duration (long) |
| 17 | DATE | Date (See Appendix C) |
| 18 | TIME_OF_DAY | Time of day (See Appendix C) |
| 19 | DATE_AND_TIME | Date and Time (See Appendix C) |
| 20 | STRING | 8-bit per character string |
| 21 | STRING2 | 16-bit per character string |
| 22 | STRINGN | N-byte per character string |
| 23 | SHORT_STRING | Short N-byte character string |
| 24 | BYTE | 8-bit string |
| 25 | DWORD | 32-bit string |
| 26 | LWORD | 64-bit string |

Obsolete data type codes shall not be valid in devices developed to this specification.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-14.2.1.3 Minimum and Maximum Value – Attributes 10 and 11

**Table 5A-14.11 Minimum and Maximum Value Semantics**

| Data Type | Description and Semantics | Minimum Value Semantics | Maximum Value Semantics | Required/ Optional/ Not Allowed |
|---|---|---|---|---|
| BYTE | Bit String – 8 bit length | The least significant bit that is set in the minimum value is the lowest valid bit to be enumerated. The most significant bit that is set in the maximum value is the highest valid bit to be enumerated. A minimum value of zero means that bit 0 is the lowest valid bit to be enumerated. A maximum value of zero is not valid.<br>Example 1 (BYTE type):<br>Min value = 0b00000001<br>Max value = 0b00001000<br>Bits 0 – 3 are enumerated.<br>Example 2 (WORD type):<br>Min value = 0b0000000000001001<br>Max value = 0b0000000000001011<br>Bits 0 – 3 are enumerated, bits 0 and 1in max value are ignored, bit 3 in min value is ignored.<br>Example 3 (DWORD type):<br>Min value = 0b00000000000000000000000000000000<br>Max value = 0b00000000000000000000000000001000<br>Bits 0 – 3 are enumerated, 0 min value implies bit 0. | | Optional |
| WORD | Bit String – 16 bit length | | | |
| DWORD | Bit String – 32 bit length | | | |
| LWORD | Bit String – 64 bit length | | | |
| STRING[1] | String<br>(2 byte length indicator, 1 byte per character) | Minimum string length | Maximum string length | Required |
| STRING2[1] | String<br>(2 byte length indicator, 2 bytes per character) | Minimum string length | Maximum string length | Required |
| STRINGN[1] | String<br>(2 byte length indicator, N bytes per character) | Minimum string length | Maximum string length | Required |
| SHORT_ STRING[1] | Character string<br>(1 byte length indicator, 1 byte characters) | Minimum string length | Maximum string length | Required |
| STRINGI[1] | International Character String (see Appendix C for data type definition) | Minimum length (in characters) of the “stringcontents” component of the string’s implicit sequence | Maximum length (in characters) of the “stringcontents” component of the string’s implicit sequence | Required |
| EPATH[1] | Encoded Path | Minimum string length | Maximum string length | Optional |
| ENGUNIT | Engineering Units | The minimum value for this data type is not defined and shall not be specified. | The maximum value for this data type is not defined and shall not be specified. | Not Allowed |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Data Type | Description and Semantics | Minimum Value Semantics | Maximum Value Semantics | Required/ Optional/ Not Allowed |
|---|---|---|---|---|
| All Other Data Types | | The minimum numeric value that may be assigned to the data value. | The maximum numeric value that may be assigned to the data value. | Optional[2] |

Table Footnotes:

1 The STRING, STRING2, STRINGN, SHORT_STRING, STRINGI and EPATH data types do not have a minimum or maximum value specification. The minimum and maximum value fields are used to present the minimum and maximum string or path lengths. In these cases, the Data Size parameter is used to represent the number of bytes required per character or encoding entry.

2 If the Minimum Value and/or Maximum Value is not specified then the minimum and/or maximum value for the parameter data value are as defined in Appendix C, Table C-2.1 based on the parameter data type.

### 5A-14.2.1.4 Scaling Factor – Attributes 13 through 21

The Scaling Factor represents actual UINT and INT parameter values in other formats. Use the following formula to determine the engineering value from the actual value:

**Equation 5A-14.1 Determining Engineering Value from Actual Value Using Scaling**

$$EngValue = \frac{(ActualValue + Offset) * Mult * Base}{\text{Div}}$$

The engineering value can then be displayed to the user in the terms specified within the Decimal Precision Instance Attribute. Use the inverse of the Scaling Formula to determine the actual value from the engineering value:

**Equation 5A-14.2 Determining Actual Value from Engineering Value Using Scaling**

$$ActualValue = \frac{(EngValue * Div)}{\text{Mult} * \text{Base}} - Offset$$

The extended precision scaling factor adds extended precision to the standard scaling factor. Use this following formula to determine the engineering value from the actual value:

**Equation 5A-14.3 Determining Engineering Value from Actual Value Using Extended Precision Scaling**

$$EngValue = \frac{(ActualValue + Offset) * Mult * Base}{\text{Div} * 10^{\text{Precision}}}$$

The engineering value can then be displayed to the user in the terms specified within the Decimal Precision attribute. Use the inverse of the extended precision scaling formula to determine the actual value from the engineering value:

**Equation 5A-14.4 Determining Actual Value from Engineering Value Using Extended Precision and Scaling**

$$ActualValue = \frac{(EngValue * Div * 10^{Precision})}{\text{Mult} * \text{Base}} - Offset$$

For the scaling formula, the following attributes are required:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-14.12 Scaling Formula Attributes**

| Attribute | Meaning |
|---|---|
| Multiplier Value | *(Mult)* Multiplier value for the scaling formula. This value can be based on another parameter. |
| Divisor Value | *(Div)* Divisor value for the scaling formula. This value can be based on another parameter specified in the Divisor Link. |
| Base Value | *(Base)* Base value for the scaling formula. This value can be based on another parameter specified in the Base Link. |
| Offset Value | *(Offset)* Offset value for the scaling formula. This value can be based on another parameter specified in the Offset Link. This attribute is an INT and can be negative. |
| Decimal Precision | (Precision) Precision value for the scaling formula. This value cannot be based on another parameter. |

Scaling values (multiplier, divisor, base, and offset) can also be based on other parameters. Scaling Links specify from which instance that scaling value is to be retrieved. If the scaling link attribute is set to 0, then that scaling value is a constant and not linked to another parameter. The following attributes specify scaling links:

**Table 5A-14.13 Scaling Links**

| Attribute | Meaning |
|---|---|
| Multiplier Link | Specifies the parameter instance from where the Multiplier Value is retrieved. |
| Divisor Link | Specifies the parameter instance from where the Divisor Value is retrieved. |
| Base Link | Specifies the parameter instance from where the Base Value is retrieved. |
| Offset Link | Specifies the parameter instance from where the Offset Value is retrieved. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-14.3 Common Services

The Parameter Object provides the following Common Services:

**Table 5A-14.14 Parameter Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0Dhex | Optional | Optional | Apply_Attributes | Causes parameter values whose use is *pending* to become actively used. |
| 0Ehex | Required | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 10hex | Optional | Required | Set_Attribute_Single | Modifies an attribute value. |
| 01hex | Optional | Conditional[4] | Get_Attributes_All | Returns a predefined listing of this objects attributes (See the Get_Attributes_All Response definition below) |
| 05hex | Optional | n/a | Reset | Resets all parameter values to the factory default. |
| 15hex | Conditional [2 3] | | Restore | Restores all parameter values from non–volatile storage. |
| | | Conditional [2] | | Restores parameter instance value from non-volatile storage. |
| 16hex | Conditional [2 3] | | Save | Saves all parameter values to non–volatile storage. |
| | | Conditional [2] | | Saves parameter instance value to non-volatile storage |
| 18hex | n/a | Conditional[1] | Get_Member | Returns the content of a selected member of an attribute. |

Table Footnotes:

1 The Get_Member service is required if any attributes with the International String (STRINGI) data type are implemented.

2 The Save and Restore services are optional. However, implementing either service requires implementation of both services.

3 If the Save and Restore services are implemented at the instance level, these services shall also be implemented at the class level.

4 Optional for Parameter Object Stubs, required for Full parameter Objects.

See the Appendix A for the definition of these common services.

### 5A-14.3.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Value portion of the Get_Attributes_All response is as follows:

**Table 5A-14.15 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 |
| 2 | UINT | Max Instance | |
| 8 | WORD | Parameter Class Descriptor | |
| 9 | UINT | Configuration Assembly Instance | |
| 10 | USINT | Native Language | 0 |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

At the **Instance level**, the setting of the Parameter Class Descriptor attribute dictates which attributes are returned by the Get_Attributes_All service:

**Table 5A-14.16 Effect of Descriptor Attribute on Get_Attributes_All Response**

| If the Parameter Class Descriptor Attribute: | Then: |
|---|---|
| does NOT have the "Supports Full Attributes" bit set, | only the implemented attributes marked *Stub* (attribute numbers 1–6) are returned by the Get_Attributes_All service |
| DOES have the "Supports Full Attributes" bit set, | The Get_Attributes_All service returns *all* implemented attributes (numbers 1–24) |

For Parameter object Stubs, the order of the attributes returned in the Attribute Value portion of the Get_Attributes_All response is as follows:

**Table 5A-14.17 Get_Attributes_All Response Data – Instance Level – Parameter Object Stub**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | Specified by Data Type Attribute (5) | Parameter Value | |
| 2 | USINT | Link Path Size | |
| 3 | Packed EPATH | Link Path | |
| 4 | WORD | Descriptor | |
| 5 | USINT | Data Type | |
| 6 | USINT | Data Size | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

For Full Parameter objects, the order of the attributes returned in the Attibute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-14.18 Get_Attributes_All Response Data – Instance Level – Full Parameter Object**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | Specified by Data Type Attribute (5) | Parameter Value | |
| 2 | USINT | Link Path Size | |
| 3 | Packed EPATH | Link Path | |
| 4 | WORD | Descriptor | |
| 5 | USINT | Data Type | |
| 6 | USINT | Data Size | |
| 7 | SHORT_STRING | Parameter Name String | |
| 8 | SHORT_STRING | Units String | |
| 9 | SHORT_STRING | Help String | |
| 10 | Specified by Data Type Attribute (5) | Minimum Value | |
| 11 | Specified by Data Type Attribute (5) | Maximum Value | |
| 12 | Specified by Data Type Attribute (5) | Default Value | |
| 13 | UINT | Scaling Multiplier | |
| 14 | UINT | Scaling Divisor | |
| 15 | UINT | Scaling Base | |
| 16 | INT | Scaling Offset | |
| 17 | UINT | Multiplier Link | |
| 18 | UINT | Divisor Link | |
| 19 | UINT | Base Link | |
| 20 | UINT | Offset Link | |
| 21 | USINT | Decimal Precision | |
| 22 | STRINGI | International Name Parameter [1] | 0 |
| 23 | STRINGI | International Engineering Units [1] | 0 |
| 24 | STRINGI | International Help String [1] | 0 |

Table Footnotes:

1. For a device that has implemented the International String attributes, the Parameter Name String character count, the Units String character count and the Help String character count shall be reported as zero. For devices that do not support the International String attributes, each (and every) International String attribute shall be reported as a single byte of zero, or may be missing from the response. If any International String attribute is supported, all three attributes shall be reported in the response.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-14.4 Object–specific Services

The Parameter Object provides the following Object–specific service:

**Table 5A-14.19 Parameter Object Specific Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| 4Bhex | n/a | Optional | Get_Enum_String | Use this service to read enumerated strings from the Parameter Instance. See below. |

Enumerated strings are human–readable strings that describe either a *bit* or a *value*, depending on the parameter's data type.

**Table 5A-14.20 Enumerated Strings Type versus Parameter Data Type**

| If the parameter's data type is: | Then the enumerated strings returned are: |
|---|---|
| BYTE, WORD, DWORD, LWORD | *Bit* enumerated strings. |
| USINT, SINT, UINT, INT, BOOL | *Value* enumerated strings. |
| All Other Data Types | Invalid for enumeration |

If the parameter's data type is BYTE, WORD, DWORD or LWORD, requesting a Get_Enum_String service with an enumerated string number of 0 on a parameter returns a SHORT_STRING that describes *bit* 0. Requesting a Get_Enum_String service with an enumerated string number of 1 on a parameter returns a SHORT_STRING that describes *bit* 1. This continues up to the maximum bit number supported by the parameter.

If the parameter's data type is SINT, USINT, INT, UINT or BOOL, requesting a Get_Enum_String service with an enumerated string number of 0 on a parameter returns a SHORT_STRING that describes the *value* of 0. Requesting a Get_Enum_String service with an enumerated string number of 1 returns a SHORT_STRING that describes the *value* of 1.

The following parameters are defined for the Get_Enum_String service:

**Table 5A-14.21 Parameters for Get_Enum_String Request**

| Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|
| Enumerated String Number | USINT | Number of Enumerated String to retrieve. | Maximum possible range of values = 0 to 255. Value indicates bit offset (relative to zero) for bit enumerations, actual value (relative to zero) for value enumerations. |

**Table 5A-14.22 Parameters for Get_Enum_String Response**

| Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|
| Enumerated String | SHORT_ STRING | Enumerated Strings | Maximum number of characters = 16 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-14.5 Behavior

The behavior of the Parameter Object is illustrated in the State Transition Diagram and State Event Matrix below.

**Figure 5A-14.2 Parameter Object State Transition Diagram**

**Table 5A-14.23 Parameter Object State Event Matrix**

| Event | State |
|---|---|
| | Existent |
| Get_Attribute_Single | Validates/service the request and returns the appropriate response. |
| Set_Attribute_Single | |
| Get_Attributes_All | |
| Reset | |
| Restore | |
| Save | |
| Get_Enum_String | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-15 Parameter Group Object

**Class Code: 10 Hex**

The Parameter Group Object identifies and provides access to groups of parameters in a device. Grouping parameters provides convenient access to related sets of parameters.

There must be one instance of this object class for each of the device's parameter groups. The instances must start at instance one and increment by one with no gaps in the instances.

Each Parameter Group Object contains a list of member Parameter Object Instances. Use the Get_Attribute service to access these parameter object instances by number.

## 5A-15.1 Class Attributes

**Table 5A-15.1 Parameter Group Object Class Attribute**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Conditional | Get | Revision | UINT | Revision of this object | See below |
| 2 | Required | Get | Max Instance | UINT | Maximum instance number of an object currently created in this class level of the device. | The largest instance number of a created object at this class hierarchy level. |
| 3 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |
| 8 | Optional | Set | Native Language | USINT | Language ID for all STRING accesses. | 0=English<br>1=French<br>2=Spanish<br>3=Italian<br>4=German<br>5=Japanese<br>6=Portugese<br>7=Mandarin Chinese |

### 5A-15.1.1 Semantics

#### 5A-15.1.1.1 Revision – Class Attribute #1

The revision of the Parameter Group Object Class can be either 1 or 2, based upon the presence of the extended static attribute field.

If the value is 01, then this attribute is OPTIONAL in implementation and the abbreviated static attribute set layout is required to exist in all instances. If the value is greater than 01, then this attribute is REQUIRED and the extended static attribute set is defined to exist within all instances.

#### 5A-15.1.1.2 Native Language – Class Attribute #8

The Native Language attribute is only used in revision 1 implementations of the Parameter Group. This attribute selects the language that will be presented in each instance's Group Name String attribute.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-15.2 Instance Attributes, Abbreviated Static Set

This attribute layout only exists when the Parameter Group revision is one.

**Table 5A-15.2 Parameter Group Object Instance Attributes, Abbreviated Static Set**

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Get | Group Name String | SHORT_ STRING | A human–readable string representing the group name (e.g., Setup, Frequency Set) | Maximum number of characters = 16 |
| 2 | Required | Get | Number of members in group | UINT | Number of parameters in group | |
| 3 | Required | Get | 1st Parameter Number in Group | UINT | Parameter Instance Number | |
| 4 | Required | Get | 2nd Parameter Number in Group | UINT | Parameter Instance Number | |
| … | | | | | | |
| n | Required | Get | (n–2)th Parameter Number in Group | UINT | Parameter Instance Number | |

The *Parameter Number* specifies the Parameter Instance of the nth member of a particular Parameter Group. Perform a Get_Attribute service on this attribute to receive the Parameter Instance Number associated with the particular group member.

## 5A-15.3 Instance Attributes, Extended Static Set

This attribute layout only exists when the Parameter Group revision is greater than one.

**Table 5A-15.3 Parameter Group Object Instance Attributes, Extended Static Set**

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Get | Group Name String | SHORT_ STRING | A human–readable string representing the group name (e.g., Setup, Frequency Set) | Maximum number of characters=16 |
| 2 | Required | Get | Number of members in group | UINT | Number of parameters in group | |
| 3 | Optional | Get | International Group Name | STRING I | One or more international language strings representing the name of the parameter group | |
| 4 – 15 | Reserved | | | | | |
| 16 | Required | Get | 1st Parameter Number in Group | UINT | Parameter Instance Number | |
| 17 | Required | Get | 2nd Parameter Number in Group | UINT | Parameter Instance Number | |
| ... | | | | | | |
| n | Required | Get | (n-15)th Parameter Number in Group | UINT | Parameter Instance Number | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-15.4 Common Services

The Parameter Group Object provides the following Common Services:

**Table 5A-15.4 Parameter Group Object Common Services**

| Service Code | Need in Implementation | | Service Name | Service Description |
|---|---|---|---|---|
| | Class | Instance | | |
| 0Ehex | Required | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 01hex | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this objects attributes (See the Get_Attributes_All Response definition below) |
| 10hex | Optional | n/a | Set_Attribute_Single | Modifies an attribute value. |
| See Appendix A for definitions of these common services. | | | | |

### 5A-15.4.1 Get_Attributes_All Response

At the **Class level,** the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-15.5 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 |
| 2 | UINT | Max Instance | |
| 8 | USINT | Native Language | 0 |

Important: A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

**Abbreviated Static Set instance** - At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response for an Abbreviated Static Set instance is as follows:

**Table 5A-15.6 Get_Attributes_All Response Data – Instance Level - Abbrev. Static Set**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | SHORT_STRING | Group Name String | |
| 2 | UINT | Number of members in group | |
| 3 | UINT | 1st Parameter Number in Group | |
| … | | | |
| 3+(n-1) | UINT | nth Parameter Number in Group | |

**Extended Static Set instance** - At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response for an Extended Static Set instance is as follows:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-15.7 Get_Attributes_All Response Data – Instance Level - Extended Static Set**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 3 | STRINGI | International Group Name | 0 |
| – | octet | Reserved[1] | |
| 2 | UINT | Number of members in group | |
| 16 | UINT | 1st Parameter Number in Group | |
| … | | | |
| 16+(n-1) | UINT | nth Parameter Number in Group | |

Table Footnotes:
1. Shall be zero.

## 5A-15.5 Object–specific Services

The Parameter Group Object provides no Object-specific services on either the Class level or Instance level.

## 5A-15.6 Behavior

The behavior of the Parameter Group Object is illustrated in the State Transition Diagram and State Event Matrix below.

**Figure 5A-15.1 Parameter Group Object State Transition Diagram**

| Event | State |
|---|---|
| | Existent |
| Get_Attribute_Single | Validates/services the request and returns the appropriate response. |
| Set_Attribute_Single | |

**Table 5A-15.8 State Event Matrix for Parameter Group Object**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-16 Group Object

**Class Code: 12 hex**

The Group Object binds other objects, typically discrete and analog, including AIP, AOP, AIG, AOG, DIP, DOP, DOG, or DIG. The objects bound to the Group must support the same attributes and services that apply to all of the bound objects (both inputs and outputs, discrete and analog) in a product.

You must establish the list of groups and/or points bound to the Group at power–up, as the Binding List is static and a Get–only attribute of the Group.

When to use the Group Object:

- when many attributes are shared among many analog and/or discrete input and output points and/or groups.
- to more efficiently access data by supporting services that may affect all members of the group.

**Figure 5A-16.1 Group Object Application**

## 5A-16.1 Class Attributes

**Table 5A-16.1 Group Object Class Attributes**

| Number | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-16.2 Instance Attributes

**Table 5A-16.2 Group Object Instance Attributes**

| Attr ID | Need in Implement | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Optional | Get | Number of Attributes | USINT | Number of attributes supported by the group | |
| 2 | Optional | Get | Attribute List | ARRAY of USINT | List of attributes supported | |
| 3 | Optional | Get | Number of Bound Instances | USINT | Number of points in a group | |
| 4 | Optional | Get | Binding | ARRAY of STRUCT: | List of instances in group | |
| | | | Class ID | UINT | | |
| | | | Instance ID | UINT | | |
| 5 | Optional | Get | Status | BOOL | Group is operating without alarms or faults | 0=good; 1=alarm state |
| 6 | Optional | Get | Owner - Vendor ID | UINT | Vendor ID of group's owner | |
| 7 | Optional | Get | Owner - Serial Number | UDINT | 32–bit serial number of group's owner | |

## 5A-16.3 Common Services

The Group Object provides the following Common Services:

**Table 5A-16.3 Group Object Common Services**

| Service Code | Need In Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0E hex | Conditional [1] | Required | Get_Attribute_Single | Returns the contents of the specified attribute |
| 01 hex | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this object's attributes (See the Get_Attributes_All_Response definition below) |

Table Footnotes:
1 The Get_Attribute_Single service is REQUIRED if any attributes are implemented.

See Appendix A for definitions of these services.

### 5A-16.3.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-16.4 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 |
| 2 | UINT | Max Instance | 0 |
| 6 | UINT | Maximum ID Number Class Attributes | 0 |
| 7 | UINT | Maximum ID Number Instance Attributes | 0 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-16.5 Get_Attributes_All Response Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | USINT | Number of Attributes | 0 |
| 2 | ARRAY of USINT | Attribute List | (null) |
| 3 | USINT | Number of Bound Instances | 0 |
| 4 | ARRAY of: | Binding List | (null) |
| | STRUCT of: | Binding | |
| | UINT | Class ID | |
| | UINT | Instance ID | |
| 5 | BOOL[1] | Status | 0 |
| 6 | UINT | Owner Vendor ID | 0 |
| 7 | UDINT | Owner Serial Number | 0 |

Table Footnotes:

1. The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

## 5A-16.4 Object–specific Services

The Group Object provides no Object-specific services.

## 5A-16.5 Behavior

The primary purpose of the Group Object is to bind Analog and Discrete Groups and/or Points. An attribute of the Group will modify the behavior or report additional status information of all objects bound to the group.

The State Transition Diagram (STD) below illustrates the Behavior of a Group Object. The states shown are equivalent to the following:

**Table 5A-16.6 Group Object State Definitions**

| This state | Is equivalent to |
|---|---|
| Non-Existent | a module without power |
| | a module that has an unrecoverable fault (e.g. processor watchdog timeout) |
| | a major reconfiguration of the module (e.g. NVS Update) |
| Binding | when groups and/or points are added or bound to the Group (executed as part of Power Up cycle) |
| Run | the point at which Get and Set attribute services can be used to access the attributes of the Group Object |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5A-16.2 Group Object State Transition Diagram**

### 5A-16.5.1 Attribute Access Rules

All attributes are gettable or settable according to their attribute access rules, but only in the Run state.

#### 5A-16.5.1.1 Status Attribute

The ***Status*** attribute is simply a logical OR of all possible failure or alarm conditions for all points bound to the group.

**Figure 5A-16.3 Group Object Status Attribute Operation**

#### 5A-16.5.1.2 Owner Vendor ID, Owner Serial Number

These attributes may be implemented in both the point and group objects. If implemented in both, the values shall be the same for the point objects and the group object that binds the point object. If a connection to this object is made using the Connection Manager, then the value for the Owner Vendor Id and Owner Serial Number shall be the Originator Vendor Id and Originator Serial Number contained in the Forward_Open request. These values shall be cleared when the connection is closed or deleted.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-17 Discrete Input Group Object

**Class Code: 1D hex**

The Discrete Input Group (DIG) Object binds a group of discrete input points in a module. All points bound to the group share all attributes contained in the group. If an attribute is shared (points have the same attributes and the same attribute values) across more than one Discrete Input Point (DIP), then that attribute can be contained in a Discrete Input Group. A Discrete Input Point can be bound to more than one Discrete Input Group.

You must establish the list of discrete input points bound to the group at power-up, as the binding list is static and is a Get–only attribute of the group.

When to use the Discrete Input Group Object:

- when a single attribute is shared among many input points.
- to more efficiently access data (services that affect all members of a group can be supported by the DIG).

**Figure 5A-17.1 Discrete Input Group Object Application**

## 5A-17.1 Class Attributes

**Table 5A-17.1 Discrete Input Group Object Class Attributes**

| Number | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-17.2 Instance Attributes

**Table 5A-17.2 Discrete Input Group Object Instance Attributes**

| Attr ID | Need in Impl | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Optional | Get | Number of Attributes | USINT | Number of attributes supported by the device | |
| 2 | Optional | Get | Attribute List | ARRAY of USINT | List of attributes supported by the device | |
| 3 | Optional | Get | Number of Bound Instances | USINT | Number of points bound to this group | |
| 4 | Optional | Get | Binding | ARRAY of:<br><br>UINT | List of all points bound to this group<br>Instance ID | Class DIP Instance #X<br>Class DIP Instance #Y... |
| 5 | Optional | Get | Status | BOOL | Status for all discrete input points in the group | 0 = OK<br>1 = Product specific alarm or status |
| 6 | Optional | Set | Off_On Delay | UINT | filter time for off to on transition 0 - 65,535 microseconds [1] | The default value is 0. |
| 7 | Optional | Set | On_Off Delay | UINT | filter time for on to off transition 0 - 65,535 microseconds [2] | The default value is 0. |

Table Footnotes:
1 The input must be on for the amount of filter time specified by the OFF_ON DELAY attribute before the ON state is recorded in the VALUE attribute.
2 The input must be off for the amount of filter time specified by the ON_OFF DELAY attribute before the OFF state is recorded in the VALUE attribute.

## 5A-17.3 Semantics

### 5A-17.3.1 On_OffDelay, Off_OnDelay

When any of these attributes are implemented at the Group level, the corresponding attribute of the point objects bound to the group shall assume the value of the Group level object attribute. The attribute shall be implemented at the point level and shall be “Get” only.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-17.4 Common Services

The Discrete Input Group Object provides the following Common Services:

**Table 5A-17.3 Discrete Input Group Object Common Services**

| Service Code | Need In Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0E hex | Conditional[1] | Required | Get_Attribute_Single | Returns the contents of the specified attribute |
| 10 hex | N/A | Optional | Set_Attribute_Single | Modifies an attribute value |
| 01 hex | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this object's attributes (See the Get_Attributes_All_Response definition below) |
| 02 hex | N/A | Optional | Set_Attributes_All | Modifies the contents of the attributes of the class or object. |

Table Footnotes:

1 The Get_Attribute_Single service is REQUIRED if any attributes are implemented.

See Appendix A for definitions of these services.

### 5A-17.4.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-17.4 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 |
| 2 | UINT | Max Instance | 0 |
| 3 | UINT | Number of Instances | 0 |
| 6 | UINT | Maximum ID Number Class Attributes | 0 |
| 7 | UINT | Maximum ID Number Instance Attributes | 0 |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-17.5 Get_Attributes_All Response Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | USINT | Number of Attributes | 0 |
| 2 | ARRAY of USINT | Attribute List | (null) |
| 3 | USINT | Number of Bound Instances | 0 |
| 4 | ARRAY of UINT | Binding | (null) |
| 5 | BOOL[1] | Status | 0 |
| 6 | UINT | Off_On Delay | 0 |
| 7 | UINT | On_Off Delay | 0 |

Table Footnotes:

1. The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

### 5A-17.4.2 Set_Attributes_All Request

No settable attributes currently exist at the **Class level** for the Discrete Input Group Object.

At the **Instance level**, the order of attributes passed in the Attribute Values portion of the Set_Attributes_All request is as follows:

**Table 5A-17.6 Set_Attributes_All Request Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 6 | UINT | Off_On Delay | |
| 7 | UINT | On_Off Delay | |

**Important:** The Set_Attributes_All service is to be supported only if **all** settable attributes shown above are implemented as settable.

## 5A-17.5 Object–specific Services

The Discrete Input Group Object does not support any Object–specific services.

## 5A-17.6 Behavior

The primary purpose of the DIG is to bind discrete input points.

An attribute of the DIG will modify the behavior or report additional status information of all DIPs bound to the group.

The State Transition Diagram below provides a graphical description of the events and corresponding state transitions. A subset of the states and events may be supported in an application, but the behavior must be consistent.

The states shown are equivalent to the following:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-17.7 Discrete Input Group Object State Definitions**

| This state | Is equivalent to |
|---|---|
| Non-Existent | a module without power |
| | a module that has an unrecoverable fault (e.g. processor watchdog timeout) |
| | a major reconfiguration of the module (e.g. NVS Update) |
| Binding | when discrete input points are added or bound to the Discrete Input Group (executed as part of Power Up cycle) |
| Run | the point at which Get and Set attribute services can be used to access the attributes of the Group Object |

**Figure 5A-17.2 State Transition Diagram for Discrete Input Group Object**

### 5A-17.6.1 Attribute Access Rules

All attributes are gettable or settable according to their attribute access rules, but only in the Run state.

#### 5A-17.6.1.1 Status Attribute

The ***Status*** attribute is simply a logical OR of all possible failure or alarm conditions for all points bound to the group.

**Figure 5A-17.3 Discrete Input Group Object Status Attribute Operation**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-18 Discrete Output Group Object

**Class Code: 1E hex**

The Discrete Output Group (DOG) Object binds a group of discrete output points in a module. All points bound to the group share all attributes contained in the group. If an attribute is shared across more than one Discrete Output Point (DOP), then it can be contained in a Discrete Output Group. A Discrete Output Point can also be bound to more than one Discrete Output Group.

You must establish the list of discrete output points bound to the group at power-up, as the binding list is static and is a Get–only attribute of the group.

When to use the Discrete Output Group Object:

- when a single attribute is shared among many output points.
- to more efficiently access data (services that affect all members of a group can be supported by the DOG).

**Figure 5A-18.1 Discrete Output Group Object Application**

## 5A-18.1 Class Attributes

**Table 5A-18.1 Discrete Output Group Object Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-18.2 Instance Attributes

**Table 5A-18.2 Discrete Output Group Object Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Optional | Get | Number of Attributes | USINT | Number of attributes supported by the device | |
| 2 | Optional | Get | Attribute List | ARRAY of USINT | List of attributes supported by the device | |
| 3 | Optional | Get | Number of Bound Instances | USINT | Number of points bound to this group | |
| 4 | Optional | Get | Binding | ARRAY of: | List of all points bound to this group | |
| | | | Instance ID | UINT | | |
| 5 | Optional | Get | Status | BOOL | Status for all discrete output points in the group | 0 = OK<br>1 = Product specific alarm or status |
| 6 | Required | Set | Command | BOOL | Changes state of all DOPs in group to Idle Mode or Run Mode | 0=idle<br>1=run |
| 7 | Optional | Set | Fault Action | BOOL | State of output after recoverable failure | 0=Fault Value attribute;<br>1=hold last state |
| 8 | Optional | Set | Fault Value | BOOL | User–defined value for use with Fault Action attribute | 0=off;<br>1=on |
| 9 | Optional | Set | Idle Action | BOOL | State of output during idle | 0=Idle Value attribute;<br>1=hold last state |
| 10 | Optional | Set | Idle Value | BOOL | User-defined value for use with Idle Action attribute | 0=off;<br>1=on |

## 5A-18.3 Semantics

### 5A-18.3.1 Command, Fault Action, Fault Value, Idle Action, Idle Value

When any of these attributes are implemented at the Group level, the corresponding attribute of the point objects bound to the group shall assume the value of the Group level object attribute. The attribute shall be implemented at the point level and shall be “Get” only.

The Idle Action and Idle Value behavior shall be the power up default behavior and the behavior when in the available state.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-18.4 Common Services

The Discrete Output Group Object provides the following Common Services:

**Table 5A-18.3 Discrete Output Group Object Common Services**

| Service Code | Need In Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0E hex | Conditional [1] | Required | Get_Attribute_Single | Returns the contents of the specified attribute |
| 10 hex | N/A | Required | Set_Attribute_Single | Modifies an attribute value |
| 01 hex | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this object's attributes (See the Get_Attributes_All_Response definition below) |
| 02 hex | N/A | Optional | Set_Attributes_All | Modifies the contents of the attributes of the class or object. |

Table Footnotes:

1 The Get_Attribute_Single service is REQUIRED if any attributes are implemented.

See Appendix A for definitions of these services.

### 5A-18.4.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-18.4 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 |
| 2 | UINT | Max Instance | 0 |
| 3 | UINT | Number of Instances | 0 |
| 6 | UINT | Maximum ID Number Class Attributes | 0 |
| 7 | UINT | Maximum ID Number Instance Attributes | 0 |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-18.5 Get_Attributes_All Response Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | USINT | Number of Attributes | 0 |
| 2 | ARRAY of USINT | Attribute List | (null) |
| 3 | USINT | Number of Bound Instances | 0 |
| 4 | ARRAY of UINT | Binding | (null) |
| 5 | BOOL[1] | Status | 0 |
| 6 | BOOL[1] | Command | |
| 7 | BOOL[1] | Fault Action | 0 |
| 8 | BOOL[1] | Fault Value | 0 |
| 9 | BOOL[1] | Idle Action | 0 |
| 10 | BOOL[1] | Idle Value | 0 |

Table Footnotes:

1. The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

### 5A-18.4.2 Set_Attributes_All Request

No settable attributes currently exist at the **Class level** for the Discrete Output Group Object.

At the **Instance level**, the order of attributes passed in the Attribute Values portion of the Set_Attributes_All request is as follows:

**Table 5A-18.6 Set_Attributes_All Request Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 7 | BOOL[1] | Fault Action | |
| 8 | BOOL[1] | Fault Value | |
| 9 | BOOL[1] | Idle Action | |
| 10 | BOOL[1] | Idle Value | |

Table Footnotes

1. The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.

**Important:** The Set_Attributes_All service is to be supported only if **all** settable attributes shown above are implemented as settable.

## 5A-18.5 Object–specific Services

The Discrete Output Group Object provides no Object–specific services.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-18.6 Behavior

The primary purpose of the Discrete Output Group Object is to bind discrete output points.

An attribute of the DOG will modify the behavior or report additional status information of all DOPs bound to the group.

The State Transition Diagram below provides a graphical description of the events and corresponding state transitions. A subset of the states and events may be supported in an application, but the behavior must be consistent.

The states shown are equivalent to the following:

**Table 5A-18.7 Discrete Output Group Object State Definitions**

| This state | Is equivalent to |
|---|---|
| Non-Existent | a module without power |
| | a module that has an unrecoverable fault (e.g. processor watchdog timeout) |
| | a major reconfiguration of the module (e.g. NVS Update) |
| Binding | when discrete output points are added or bound to the Discrete Output Group (executed as part of Power Up cycle) |
| Run | the point at which Get and Set attribute services can be used to access the attributes of the Group Object |

**Figure 5A-18.2 Discrete Output Group Object State Transition Diagram**

### 5A-18.6.1 Attribute Access Rules

All attributes are gettable or settable according to their attribute access rules, but only in the Run state.

#### 5A-18.6.1.1 Status Attribute

The ***Status*** attribute is simply a logical OR of all possible failure or alarm conditions for all points bound to the group.

**Figure 5A-18.3 Discrete Output Group Object Status Attribute Operation**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-19 Discrete Group Object

**Class Code: 1F hex**

The Discrete Group (DG) Object binds other discrete objects including DIP, DOP, DIG or DOG. The objects (both input and output) bound to the Discrete Group must support the same attributes and services.

You must establish the list of discrete groups and/or points bound to the Discrete Group at power–up, as the Binding List is static and a Get–only attribute of the Discrete Group.

When to use the Discrete Group Object:

- when a single attribute is shared among many discrete input and output points and/or groups.
- to more efficiently access data (services that affect all members of the group can be supported by the DG).

**Figure 5A-19.1 Discrete Group Object Application**

For example:

The Discrete Group Object could be used when a module contains 4 DIPs and 4 DOPs; the DIPs are grouped into a DIG to share a common attribute, and the DOPs are grouped into a DOG to share a common output behavior.

You want all points to share a common status. The DG has the common attribute Status and binds the DIG and DOG, which in turn bind the individual points.

## 5A-19.1 Class Attributes

**Table 5A-19.1 Discrete Group Object Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-19.2 Instance Attributes

**Table 5A-19.2 Discrete Group Object Instance Attributes**

| Attr ID | Need in Impl | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Optional | Get | Number of Attributes | USINT | Number of attributes supported by the group | |
| 2 | Optional | Get | Attribute List | ARRAY of USINT | List of attributes supported | |
| 3 | Optional | Get | Number of Bound Instances | USINT | Number of points in a group | |
| 4 | Optional | Get | Binding | ARRAY of STRUCT: | List of all instances group | |
| | | | Class ID | UINT | | |
| | | | Instance ID | UINT | | |
| 5 | Optional | Get | Status | BOOL | Group is operating without alarms or faults | 0=good;<br>1=alarm state |

## 5A-19.3 Common Services

The Discrete Group Object provides the following Common Services:

**Table 5A-19.3 Discrete Group Object Common Services**

| Service Code | Need In Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0E hex | Conditional [1] | Required | Get_Attribute_Single | Returns the contents of the specified attribute |
| 01 hex | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this object's attributes (See the Get_Attributes_All_Response definition below) |

Table Footnotes:

1 The Get_Attribute_Single service is REQUIRED if any attributes are implemented.

See Appendix A for definitions of these services.

### 5A-19.3.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-19.4 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 |
| 2 | UINT | Max Instance | 0 |
| 3 | UINT | Number of Instances | 0 |
| 6 | UINT | Maximum ID Number Class Attributes | 0 |
| 7 | UINT | Maximum ID Number Instance Attributes | 0 |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-19.5 Get_Attributes_All Response Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | USINT | Number of Attributes | 0 |
| 2 | ARRAY of USINT | Attribute List | (null) |
| 3 | USINT | Number of Bound Instances | 0 |
| 4 | ARRAY of: | Binding List | (null) |
| | STRUCT of: | Binding | |
| | UINT | Class ID | |
| | UINT | Instance ID | |
| 5 | BOOL[1] | Status | 0 |

Table Footnotes:

1. The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

## 5A-19.4 Object–specific Services

The Discrete Group Object provides no Object–specific services

## 5A-19.5 Behavior

The primary purpose of the Discrete Group is to bind Discrete Groups and/or Points.

An attribute of the DG modifies the behavior or reports additional status information of all objects bound to the group.

The State Transition Diagram below illustrates the behavior of a Discrete Group.

**Table 5A-19.6 Discrete Group Object State Definitions**

| This state | Is equivalent to |
|---|---|
| Non-Existent | a module without power |
| | a module that has an unrecoverable fault (e.g. processor watchdog timeout) |
| | a major reconfiguration of the module (e.g. NVS Update) |
| Binding | when the discrete groups and/or points are added or bound to the Discrete Group (executed as part of Power Up cycle) |
| Run | the point at which Get and Set attribute services can be used to access the attributes of the Discrete Group Object |

**Figure 5A-19.2 State Transition Diagram for Discrete Group Object**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-19.5.1 Attribute Access Rules

All attributes are gettable or settable according to their attribute access rules, but only in the Run state.

#### 5A-19.5.1.1 Status Attribute

The ***Status*** attribute is simply a logical OR of all possible failure or alarm conditions for all points bound to the group.

**Figure 5A-19.3 Discrete Group Object Status Attribute Function**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-20 Analog Input Group Object

**Class Code: 20 hex**

The Analog Input Group Object (AIG) binds a group of analog input points in a module. All points bound to the group share all attributes contained in the group . If an attribute is shared (points have the same attributes AND the same attribute values) across more than one Analog Input Point (AIP), then that attribute can be contained in an Analog Input Group. An Analog Input Point can be bound to more than one Analog Input Group.

You must establish the list of analog input points bound to the group at power-up, as the binding list is static and is a Get–only attribute of the group.

When to use the Analog Input Group Object:

- when a single attribute is shared among many input points.
- to more efficiently access data (services that affect all members of the group can be supported by the AIG).

**Figure 5A-20.1 Analog Input Group Object Application**

## 5A-20.1 Class Attributes

**Table 5A-20.1 Analog Input Group Object Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-20.2 Instance Attributes

**Table 5A-20.2 Analog Input Group Object Instance Attributes**

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Optional | Get | Number of Attributes | USINT | Number of attributes supported by the group | |
| 2 | Optional | Get | Attribute List | ARRAY of USINT | List of attributes supported | |
| 3 | Optional | Get | Number of Bound Instances | USINT | Number of points in a group | |
| 4 | Optional | Get | Bound Instances | ARRAY of UINT | List of all points bound to this group | |
| 5 | Optional | Get | Status | BOOL | Group is operating without alarms or faults | 0=good;<br>1=alarm state |
| 6 | Optional | Get | Owner - Vendor ID | UINT | Vendor ID of group's owner | |
| 7 | Optional | Get | Owner - Serial Number | UDINT | 32–bit serial number of group's owner | |
| 8 | Optional | Set | Value Data Type | USINT | Determines the data type of AIP's Value | 0 = INT<br>1 = REAL<br>2 = USINT<br>3 = SINT<br>4 = DINT<br>5 = LINT<br>6 = UINT<br>7 = UDINT<br>8 = ULINT<br>9 = LREAL<br>100=vendor specific |
| 9 | Reserved for CIP | | | | | |
| 10 | Optional | Set | Temp Mode | BOOL | Temperature scale to use when reporting a temperature value | 0 = Celsius<br>1 = Fahrenheit |

## 5A-20.3 Semantics

### 5A-20.3.1 Value Data Type

When this attribute is implemented at the Group level, the corresponding attribute of the point objects bound to the group shall assume the value of the Group level object attribute.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-20.4 Common Services

The Analog Input Group Object provides the following Common Services:

**Table 5A-20.3 Analog Input Group Object Common Services**

| Service Code | Need In Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0E hex | Conditional [1] | Required | Get_Attribute_Single | Returns the contents of the specified attribute |
| 10 hex | N/A | Required [2] | Set_Attribute_Single | Modifies an attribute value |
| 01 hex | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this object's attributes (See the Get_Attributes_All Response definition below) |
| 02 hex | N/A | Optional | Set_Attributes_All | Modifies the contents of the attributes of the class or object. |

Table Footnotes:
1 The Get_Attribute_Single service is REQUIRED if any attributes are implemented.
2 The Set_Attribute_Single service is required only if Instance Attributes #8 and/or #10 are implemented.

See Appendix A for definitions of these services.

### 5A-20.4.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-20.4 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 |
| 2 | UINT | Max Instance | 0 |
| 6 | UINT | Maximum ID Number Class Attributes | 0 |
| 7 | UINT | Maximum ID Number Instance Attributes | 0 |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-20.5 Get_Attributes_All Response Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | USINT | Number of Attributes | 0 |
| 2 | ARRAY of USINT | Attribute List | (null) |
| 3 | USINT | Number of Bound Instances | 0 |
| 4 | ARRAY of UINT | Bound Instances | (null) |
| 5 | BOOL[1] | Status | 0 |
| 6 | UINT | Owner Vendor ID | 0 |
| 7 | UDINT | Owner Serial Number | 0 |
| 8 | USINT | Value Data Type | 0 |
| 10 | BOOL[1] | Temp Mode | 0 |

Table Footnotes

1. The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

### 5A-20.4.2 Set_Attributes_All Request

No settable attributes currently exist at the **Class level** for the Analog Input Group Object.

At the **Instance level**, the order of attributes passed in the Attribute Values portion of the Set_Attributes_All request is as follows:

**Table 5A-20.6 Set_Attributes_All Request Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 8 | USINT | Value Data Type | |
| 10 | BOOL[1] | Temp Mode | |

Table Footnotes

1. The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.

**Important:** The Set_Attributes_All service is to be supported only if **all** settable attributes shown above are implemented as settable.

## 5A-20.5 Object–specific Services

The Analog Input Group Object provides no Object–specific services.

## 5A-20.6 Behavior

The primary purpose of the Analog Input Group Object is to bind analog input points.

An attribute of the AIG modifies the behavior or reports additional status information of all AIPs bound to the group.

The State Transition Diagram below provides a graphical description of the events and corresponding state transitions. A subset of the states and events may be supported in an application, but the behavior must be consistent.

The states shown are equivalent to the following:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-20.7 Analog Input Group Object State Definitions**

| This state | Is equivalent to |
|---|---|
| Non-Existent | a module without power |
| | a module that has an unrecoverable fault (e.g. processor watchdog timeout) |
| | a major reconfiguration of the module (e.g. NVS Update) |
| Binding | when analog input points are added or bound to the Analog Input Group (executed as part of Power Up cycle) |
| Run | the point at which Get and Set attribute services can be used to access the attributes of the Group Object |

**Figure 5A-20.2 State Transition Diagram for Analog Input Group Object**

## 5A-20.6.1 Attribute Access Rules

All attributes are gettable or settable according to their attribute access rules.

### 5A-20.6.1.1 Status Attribute

The ***Status*** attribute is simply a logical OR of all possible failure or alarm conditions for all points bound to the group.

**Figure 5A-20.3 Analog Input Group Object Status Attribute Function**

### 5A-20.6.1.2 Value Data Type Attribute

The ***Value Data Type*** attribute determines the data type to be used by the attribute Value (and perhaps other attributes as well). If Value Data Type is not used, then Value defaults to INT. Value (and other attributes) may behave as REAL, USINT, or any other data type based upon the definition of Value Data Type, which ultimately provides the ability for the analog point to be defined in any length necessary.

### 5A-20.6.1.3 Owner Vendor ID and Owner Serial Number Attributes

These attributes may be implemented in both the point and group objects. If implemented in both, the values shall be the same for the point objects and the group object that binds the point object. If a connection to this object is made using the Connection Manager, then the value for the Owner Vendor Id and Owner Serial Number shall be the Originator Vendor Id and Originator Serial Number contained in the Forward_Open request. These values shall be cleared when the connection is closed or deleted.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-20.6.1.4 Temp Mode Attribute

The attribute Temp Mode is a Boolean data type that governs the behavior of the Value attribute of the AIP when returning a temperature. The temperature is returned in either Celcius or Fahrenheit according to the current value of the Temp Mode attribute. AIP values are typically returned as a temperature for thermocouple and RTD channels when linearization is turned on.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-21 Analog Output Group Object

**Class Code: 21 hex**

The Analog Output Group Object (AOG) binds a group of analog output points in a module. All points bound to the group share all attributes contained in the group. If an attribute is shared (points have the same attributes and the same attribute values) across more than one Analog Output Point (AOP), then that attribute can be contained in an Analog Output Group. An Analog Output Point can be bound to more than one Analog Output Group.

You must establish the list of analog output points bound to the group at power-up, as the binding list is static and is a Get–only attribute of the group.

When to use the Analog Output Group Object:

- when a single attribute is shared among many input points.
- to more efficiently access data (services that affect all members of a group can be supported by the AOG).

**Figure 5A-21.1 Analog Output Group Object Application**

## 5A-21.1 Class Attributes

**Table 5A-21.1 Analog Output Group Object Class Attributes**

| Number | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Get | Revision | UINT | Revision of this object | The current value assigned to this attribute is two (02) |
| 2 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-21.2 Instance Attributes

**Table 5A-21.2 Analog Output Group Object Instance Attributes**

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Optional | Get | Num Attributes | USINT | Number of attributes supported by the group | |
| 2 | Optional | Get | Attribute List | Array of USINT | List of attributes supported by the group | |
| 3 | Optional | Get | Number of Bound Instances | USINT | Number of bindings in the group | |
| 4 | Required if any Analog Output Points are not included in the Analog Output Group | Get | Bound Instances | ARRAY of UINT | List of all points bound to this group | |
| 5 | Optional | Get | Status | BOOL | Group is operating without alarms or faults | 0 = good;<br>1 = alarm state |
| 6 | Optional | Get | Owner - Vendor ID | UINT | Vendor ID of group's owner | |
| 7 | Optional | Get | Owner - Serial Number | UDINT | 32–bit serial number of group's owner | |
| 8 | Optional | Set | Value Data Type | USINT | Determines the data type of any bound Values | Default = 0 = INT<br>0 = INT<br>1 = REAL<br>2 = USINT<br>3 = SINT<br>4 = DINT<br>5 = LINT<br>6 = UINT<br>7 = UDINT<br>8 = ULINT<br>9 = LREAL<br>100=vendor specific |
| 9 | Required | Set | Command | BOOL | Changes state of AOP to Idle Mode or Run Mode | 0 = idle<br>1 = run |
| 10 | Optional | Set | Fault Action | USINT | Output value to go to on failure or fault | 0 = use Fault Value [1]<br>1 = hold last state (default)<br>2 = low limit<br>3 = high limit<br>4 – 99 = reserved<br>100 – 199 = vendor specific<br>200 – 255 = reserved |
| 11 | Required if Fault Action is Implemented | Set | Fault Value | INT or based on attribute 8 | User–defined value for use with Fault Action attribute | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 12 | Optional | Set | Idle Action | USINT | Output value to go to on Idle | 0 = use Idle Value[1]<br>1 = hold last state (default)<br>2 = low limit<br>3 = high limit<br>4 – 99 = reserved<br>100 – 199 = vendor specific<br>200 – 255 = reserved |
| 13 | Required if Idle Action is implemented | Set | Idle Value | INT or based on attribute 8 | User-defined value for use with Idle Action attribute | |

Table Footnotes:
1 All values 0, 1, 2, and 3 shall be supported when the Fault Action or Idle Action attributes are implemented.

## 5A-21.3 Semantics

### 5A-21.3.1 Command, Fault Action, Fault Value, Idle Action, Idle Value, Value Data Type

When any of these attributes are implemented at the Group level, the corresponding attribute of the point objects bound to the group shall assume the value of the Group level object attribute. The attribute shall be implemented at the point level and shall be “Get” only.

The Idle Action and Idle Value behavior shall be the power up default behavior and the behavior when in the available state.

### 5A-21.3.2 Value Data Type

When this attribute is implemented at the Group level, the corresponding attribute of the point objects bound to the group shall assume the value of the Group level object attribute. The attribute shall be implemented at the point level and shall be “Get” only.

## 5A-21.4 Common Services

The Analog Output Group Object provides the following Common Services:

**Table 5A-21.3 Analog Output Group Object Common Services**

| Service Code | Need In Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0E hex | Conditional [1] | Required | Get_Attribute_Single | Returns the contents of the specified attribute |
| 10 hex | N/A | Required [2] | Set_Attribute_Single | Modifies an attribute value |
| 01 hex | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this object’s attributes (See the Get_Attributes_All_Response definition below) |
| 02 hex | N/A | Optional | Set_Attributes_All | Modifies the contents of the attributes of the class or object. |

Table Footnotes:
1 The Get_Attribute_Single service is REQUIRED if any attributes are implemented.
2 The Set_Attribute_Single service is required only if Instance Attributes #8 and/or #10 are implemented.

See Appendix A for definitions of these services.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-21.4.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-21.4 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 [1] |
| 2 | UINT | Max Instance | 0 |
| 6 | UINT | Maximum ID Number Class Attributes | 0 |
| 7 | UINT | Maximum ID Number Instance Attributes | 3 |

Table Footnotes:

1. The default value only applies for Revision 1 of this object. Newer revisions require that this attribute be implemented, which will override this default value.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-21.5 Get_Attributes_All Response Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 9 | BOOL[1] | Command | |
| 1 | USINT | Number of Attributes | 0 |
| 2 | ARRAY of USINT | Attribute List | (null) |
| 3 | USINT | Number of Bound Instances | 0 |
| 4 | ARRAY of UINT | Bound Instances | (null) |
| 5 | BOOL[1] | Status | 0 |
| 6 | UINT | Owner Vendor ID | 0 |
| 7 | UDINT | Owner Serial Number | 0 |
| 8 | USINT | Value Data Type | 0 |
| 10 | USINT | Fault Action | 1 |
| 11 | Specified by Value Data Type (Attribute #8) | Fault Value | 0 |
| 12 | USINT | Idle Action | 1 |
| 13 | Specified by Value Data Type (Attribute #8) | Idle Value | 0 |

Table Footnotes:

1. The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-21.4.2 Set_Attributes_All Request

No settable attributes currently exist at the **Class level** for the Analog Output Group Object.

At the **Instance level**, the order of attributes passed in the Attribute Values portion of the Set_Attributes_All request is as follows:

**Table 5A-21.6 Set_Attributes_All Request Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 9 | BOOL[1] | Command | |
| 8 | USINT | Value Data Type | |
| 10 | USINT | Fault Action | |
| 11 | Specified by Value Data Type (Attribute #8) | Fault Value | |
| 12 | USINT | Idle Action | |
| 13 | Specified by Value Data Type (Attribute #8) | Idle Value | |

Table Footnotes:

1. The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.

**Important:** The Set_Attributes_All service is to be supported only if **all** settable attributes shown above are implemented as settable.

## 5A-21.5 Object–specific Services

The Analog Output Group Object provides no Object–specific services.

## 5A-21.6 Behavior

The primary purpose of the Analog Output Group is to bind Analog Output Points.

An attribute of the AOG will modify the behavior or report additional status information of all AOPs bound to the group.

The State Transition Diagram below provides a graphical description of the events and corresponding state transitions. A subset of the states and events may be supported in an application, but the behavior must be consistent.

The states shown are equivalent to the following:

**Table 5A-21.7 Analog Output Group Object State Definitions**

| This state | Is equivalent to |
|---|---|
| Non-Existent | A module without power |
| | A module that has an unrecoverable fault (e.g. processor watchdog timeout) |
| | A major reconfiguration of the module (e.g. NVS Update) |
| Binding | When analog output points are added or bound to the Analog Output Group (executed as part of Power Up cycle) |
| Run | The point at which Get and Set attribute services can be used to access the attributes of the Group Object |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5A-21.2 State Transition Diagram for Analog Output Group Object**

### 5A-21.6.1 Attribute Access Rules

All attributes are gettable or settable according to their attribute access rules, but only in the Run state.

#### 5A-21.6.1.1 Status Attribute

The ***Status*** attribute is simply a logical OR of all possible failure or alarm conditions for all points bound to the group.

**Figure 5A-21.3 Analog Output Group Object Status Attribute Function**

#### 5A-21.6.1.2 Value Data Type Attribute

The ***Value Data Type*** attribute determines the data type to be used by the attribute Value (and perhaps other attributes as well). If Value Data Type is not used, then Value defaults to INT. Value (and other attributes) may behave as REAL, USINT, or any other data type based upon the definition of Value Data Type, which ultimately provides the ability for the analog point to be defined in any length necessary.

#### 5A-21.6.1.3 Owner Vendor ID and Owner Serial Number Attributes

These attributes may be implemented in both the point and group objects. If implemented in both, the values shall be the same for the point objects and the group object that binds the point object. If a connection to this object is made using the Connection Manager Object, then the value for the Owner Vendor Id and Owner Serial Number shall be the Originator Vendor Id and Originator Serial Number contained in the Forward_Open request. These values shall be cleared when the connection is closed or deleted.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-21.6.1.4 Fault Action/Idle Action Values for Analog Output Group and Analog Output Point

**Table 5A-21.8 Corresponding Attribute Values for Output Group/Analog Output Point Objects**

| Value of Analog Output Group Instance Attributes #10 and #12 | Meaning | Value of Corresponding Analog Output Point Instance Attributes #9 and #10 |
|---|---|---|
| 0 | Use Fault/Idle Value | 3 |
| 1 | Hold Last State (default) | 0 |
| 2 | Use Low Limit | 1 |
| 3 | Use High Limit | 2 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-22 Analog Group Object

**Class Code: 22 hex**

The Analog Group (AG) Object binds other analog objects including AIP, AOP, AIG or AOG. The objects (both inputs and outputs) bound to the Analog Group must support the same attributes and services.

You must establish the list of analog groups and/or points bound to the Analog Group at power–up, as the Binding List is static and a Get–only attribute of the Analog Group.

When to use the Analog Group Object:

- when a single attribute is shared among many analog input and output points and/or groups.
- to more efficiently access data (services that affect all members of a group and/or points can be supported by the AG).

**Figure 5A-22.1 Anaog Group Object Application**

## 5A-22.1 Class Attributes

**Table 5A-22.1 Analog Group Object Class Attributes**

| Number | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-22.2 Instance Attributes

**Table 5A-22.2 Analog Group Object Instance Attributes**

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Optional | Get | Number of Attributes | USINT | Number of attributes supported by the group | |
| 2 | Optional | Get | Attribute List | ARRAY of USINT | List of attributes supported | |
| 3 | Optional | Get | Number of Bound Instances | USINT | Number of points in a group | |
| 4 | Optional | Get | Binding | ARRAY of STRUCT: | List of all instances in group | |
| | | | Class ID | UINT | | |
| | | | Instance ID | UINT | | |
| 5 | Optional | Get | Status | BOOL | Group is operating without alarms or faults | 0=good;<br>1=alarm state |
| 6 | Optional | Get | Owner - Vendor ID | UINT | Vendor ID of group's owner | |
| 7 | Optional | Get | Owner - Serial Number | UDINT | 32–bit serial number of group's owner | |
| 8 | Optional | Set | Value Data Type | BYTE | Determines the data type of all bound object's Value which propagate through another binding if groups are bound to AG | 0 = INT<br>1 = REAL<br>2 = USINT<br>3 = SINT<br>4 = DINT<br>5 = LINT<br>6 = UINT<br>7 = UDINT<br>8 = ULINT<br>9 = LREAL<br>100 = vendor specific |

## 5A-22.3 Common Services

The Analog Group Object provides the following Common Services:

**Table 5A-22.3 Analog Group Object Common Services**

| Service Code | Need In Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0E hex | Conditional [1] | Required | Get_Attribute_Single | Returns the contents of the specified attribute |
| 10 hex | N/A | Required [2] | Set_Attribute_Single | Modifies an attribute value |
| 01 hex | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this object's attributes (See the Get_Attributes_All_Response definition below) |

Table Footnotes:

1 The Get_Attribute_Single service is REQUIRED if any attributes are implemented.

2 The Set_Attribute_Single service is required only if Instance Attributes #8 and/or #10 are implemented.

See Appendix A for definitions of these services.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-22.3.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-22.4 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 |
| 2 | UINT | Max Instance | 0 |
| 6 | UINT | Maximum ID Number Class Attributes | 0 |
| 7 | UINT | Maximum ID Number Instance Attributes | 0 |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-22.5 Get_Attributes_All Response Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | USINT | Number of Attributes | 0 |
| 2 | ARRAY of USINT | Attribute List | (null) |
| 3 | USINT | Number of Bound Instances | 0 |
| 4 | ARRAY of: | Binding List | (null) |
| | STRUCT of: | Binding | |
| | UINT | Class ID | |
| | UINT | Instance ID | |
| 5 | BOOL[1] | Status | 0 |
| 6 | UINT | Owner Vendor ID | 0 |
| 7 | UDINT | Owner Serial Number | 0 |
| 8 | BYTE | Value Data Type | 0 |

Table Footnotes:

1. The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

## 5A-22.4 Object–specific Services

The Analog Group Object provides no Object–specific services.

## 5A-22.5 Behavior

The primary purpose of the Analog Group Object is to bind analog groups and/or points.

An attribute of the AG modifies the behavior or reports additional status information of all objects bound to the group.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The State Transition Diagram below provides a graphical description of the events and corresponding state transitions. A subset of the states and events may be supported in an application, but the behavior must be consistent.

Except for highly configurable modules (which could have explicit Create and Delete events), the states shown in the STD are equivalent to the following:

**Table 5A-22.6 Analog Group Object State Definitions**

| This state | Is equivalent to |
|---|---|
| Non-Existent | a module without power<br>a module that has an unrecoverable fault (e.g. processor watchdog timeout)<br>a major reconfiguration of the module (e.g. NVS Update) |
| Binding | when the analog input and/or output groups and/or points are added or bound to the Analog Group (executed as part of Power Up cycle) |
| Run | the point at which Get and Set attribute services can be used to access the attributes of the Analog Group |

**Figure 5A-22.2 Analog Group Object State Transition Diagram**

### 5A-22.5.1 Attribute Access Rules

All attributes are gettable or settable according to their attribute access rules, but only in the Run state.

#### 5A-22.5.1.1 Status Attribute

The ***Status*** attribute is simply a logical OR of all possible failure or alarm conditions for all points bound to the group.

**Figure 5A-22.3 Analog Group Object Status Attribute Function**

#### 5A-22.5.1.2 Value Data Type Attribute

The ***Value Data Type*** attribute determines the data type to be used by all of the bound AIP's Value attribute (and perhaps other attributes as well). If Value Data Type is not used, then Value defaults to INT. Value (and other attributes) may behave as REAL, USINT, or any other data type based upon the definition of Value Data Type, which ultimately provides the ability for the analog point to be defined in any length necessary.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5A-22.5.1.3 Owner Vendor ID and Owner Serial Number Attributes

These attributes may be implemented in both the point and group objects. If implemented in both, the values shall be the same for the point objects and the group object that binds the point object. If a connection to this object is made using the Connection Manager, then the value for the Owner Vendor Id and Owner Serial Number shall be the Originator Vendor Id and Originator Serial Number contained in the Forward_Open request. These values shall be cleared when the connection is closed or deleted.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-23 Position Sensor Object

## Class Code: 23 hex

The Position Sensor Object models an absolute position sensor in a product. Behaviors in the object extend the basic position sensor capability to include zero offset, and position boundary checking (CAM switch).

The Position Sensor Object interface is to real position sensor hardware such as an absolute digital encoder, an analog resolver or other absolute position-input device.

## 5A-23.1 Revision History

Since the initial release of this object class definition changes have been made that require a revision update of this object class. The table below represents the revision history.

**Table 5A-23.1 Revision History**

| Revision | Reason for Change |
|---|---|
| 01 | Initial release of this object class |
| 02 | Add new attributes |

## 5A-23.2 Class Attributes

**Table 5A-23.2 Position Sensor Object Class Attributes**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Get | Revision | UINT | Revision of this object | The current value assigned to this attribute is 02. |
| 2 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |

## 5A-23.3 Instance Attributes

**Table 5A-23.3 Position Sensor Object Instance Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Optional | Get | NV | Number of Attributes | USINT | Number of attributes supported in this product | |
| 2 | Optional | Get | NV | Attribute List | Array of USINT | List of attributes supported in this product. | |
| 3 | Conditional [1] | Get | V | Position Value Unsigned | UDINT | Current position. | Physical position See Semantics Section |
| 4 | Optional | Get | V | CAM | BOOL | Virtual CAM switch value | 0 = Off<br>1 = On |
| 5 | Optional | Set | V | Value Bit Resolution | USINT | Position sensor resolution. | See Semantics. |
| 6 | Optional | Set | V | Zero Offset | UDINT | Value attribute zero offset | See Semantics. |
| 7 | Optional | Set | V | CAM Low Limit | UDINT | Virtual CAM switch low limit. | The default is 0 |
| 8 | Optional | Set | V | CAM High Limit | UDINT | Virtual CAM switch high limit. | The default is 0 |
| 9 | Optional | Set | V | Auto Zero | BOOL | Auto zero control. | Rising edge sets Zero Offset to Position Value |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Position Sensor Object, Class Code: $23_{Hex}$

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 10 | Conditional[1] | Get | V | Position Value Signed | DINT | Current position | The content is based upon 14, 15, 16, and 17. |
| 11 | Optional | Get | NV | Position Sensor Type | UINT | Specifies the device type | See Semantics section |
| 12 | Required | Set | NV | Direction Counting Toggle | BOOL | Defines the direction of increasing 'Position Value' | Default = 0.<br>See semantics section |
| 13 | Optional | Set | NV | Commissioning Diagnostic Control | BOOL | Check encoder at encoder stand still | 0 = OFF<br>1 = ON (Default) |
| 14 | Optional | Set | NV | Scaling Function Control | BOOL | *Physical resolution* span (attribute 42) is converted to a numerical value | 0 = OFF<br>1 = ON (Default) |
| 15 | Optional | Set | NV | Position Format | ENGUNIT | Format of the position value of other attributes. | Supported units:<br>counts(default )<br>Millimeter<br>micron<br>nanometer<br>inch<br>thousandths inch<br>ten thousandths inch<br>0x0800-0x0FFF = vendor specific |
| 16 | Optional | Set | NV | Measuring Units per Span | UDINT | Number of distinguishable steps per one complete span. Less than or equal to *Physical Resolution Span* (attribute 42). | For rotary devices a span equals one revolution. |
| 17 | Optional | Set | NV | Total Measuring Range in Measuring Units | UDINT | Steps over the total measuring range. Only used for rotary encoders. | |
| 18 | Optional | Set | NV | Position Measuring Increment | UDINT | Specifies the smallest incremental change of the *Position Value Signed* or *Position Value Unsigned* attributes | Units depend on *Position Format* attribute.<br>Default = 1. |
| 19 | Optional | Set | NV | Preset Value | DINT | Output position value is set to Preset Value. | See Semantics section |
| 20 | Optional | Set | NV | COS/delta | UDINT | Value for position change in COS mode. | See Semantics section |
| 21 | Optional | Get | V | Position State Register | BYTE | The state of software limit switch. | Bit 0: 1 = Out of Range<br>Bit 1: 1 = Range overflow<br>Bit 2: 1 = Range underflow |
| 22 | Optional | Set | NV | Position Low Limit | DINT | Low Limit Position | See Semantics section |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Position Sensor Object, Class Code: $23_{Hex}$

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 23 | Optional | Set | NV | Position High Limit | DINT | High Limit Position<br>When attribute 10 is greater than this value, the *Position State Register*, bit 1 shall be set to one (1). | See Semantics section |
| 24 | Optional | Get | V | Velocity Value | DINT | Current speed where the format of this value is defined in attributes 25 & 26. | Default is attribute 15 per second<br>The meaning of the sign is affected by the value of attribute 12. See Semantics section |
| 25 | Conditional | Set | NV | Velocity Format | ENGUINT | Format of the velocity attributes. | 0x1f04 = counts (Steps) per second (default)<br>See Semantics section |
| 26 | Conditional | Set | NV | Velocity Resolution | UDINT | Specifies the smallest incremental change of the *Velocity Value* attribute 24. | Units depend on Attr 25<br>Default = 1.<br>See Semantics section |
| 27 | Conditional | Set | NV | Minimum Velocity Setpoint | DINT | Value for minimum velocity trigger threshold. Affects Min. Velocity Flag in status *Warning* Attribute 47 | Default = 0x80000000.<br>See Semantics section |
| 28 | Conditional | Set | NV | Maximum Velocity Setpoint | DINT | Value for maximum velocity trigger threshold. Affects Max. Velocity Flag in status *Warning* Attribute 47 | See Semantics section<br>Default = 0xEFFFFFFF.<br>See Semantics section |
| 29 | Optional | Get | V | Acceleration Value | DINT | Current Acceleration where the format of this value is defined in attributes 30 & 31. | Default is *Velocity Value* attribute 24 per second$^2$<br>Positive value is acceleration, negative is deceleration.<br>See Semantics section |
| 30 | Conditional | Set | NV | Acceleration Format | ENGUINT | Format of the Acceleration attributes. | 0x1500 = m/s$^2$ (default)<br>See Semantics section |
| 31 | Conditional | Set | NV | Acceleration Resolution | UDINT | Specifies the smallest incremental change of the *Acceleration Value*, attribute 29 | Default = 1.<br>See Semantics section |
| 32 | Conditional | Set | NV | Minimum Acceleration Setpoint | DINT | Value for minimum acceleration trigger threshold. | Default = 0x8000 0000.<br>See Semantics section |
| 33 | Conditional | Set | NV | Maximum Acceleration Setpoint | DINT | Value for maximum acceleration trigger threshold. | Default = 0xEFFF FFFF .<br>See Semantics section |
| 34 | Optional | Get | NV | Number of CAM Channels | USINT | Contains the number of independent cams | See Semantics section |
| 35 | Conditional | Get | V | CAM Channel State register | ARRAY of BOOL | Contains state of independent cam channels | Bit 0 = CAM_ 1<br>State ...<br>Bit x = CAM x+1 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Position Sensor Object, Class Code: $23_{Hex}$

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 36 | Conditional | Set | NV | CAM Channel Polarity Register | ARRAY of BOOL | Determines the polarity for each *Cam Channel State Register* | Bit 0 = CAM_ 1 Polarity ... Bit x = CAM x+1 |
| 37 | Conditional | Set | NV | CAM Channel Enable Register | ARRAY of BOOL | Enables each independent cam channel where size of array is equal to *Number of CAM Channels* attribute. | Bit 0 = CAM_ 1 Enable ... Bit x = CAM x+1 |
| 38 | Conditional | Set | NV | CAM Low Limit | ARRAY of DINT | Switch point for the lower limit where size of array is equal to *Number of CAM channels* attribute. When *Position Value Signed* is less than *CAM Low Limit* value, then *CAM Channel State Register* = 1. | |
| 39 | Conditional | Set | NV | CAM High Limit | ARRAY of DINT | Switch point for the higher limit where size of array is equal to "Number of CAM channel" *Number of CAM channels* attribute. *Position Value Signed* is greater than *CAM High Limit* value, then *CAM Channel State Register* = 1. | |
| 40 | Conditional | Set | NV | CAM Hysteresis | ARRAY of UINT | This value will be added to the *CAM High Limit* and subtracted from the *CAM Low Limit* when calculating the cam state. | See Semantics section |
| 41 | Optional | Get | V | Operating Status | BYTE | Encoder diagnostic operating status | See Semantics Section |
| 42 | Optional | Get | NV | Physical Resolution Span | UDINT | Number of distinguishable steps per one complete span. | For rotary devices, a span equals one revolution. |
| 43 | Optional | Get | NV | Number of Spans | UINT | This is equal to the number of turns when a rotary type device is used. | Default = 1. |
| 44 | Optional | Get | V | Alarms | WORD | Indicates a malfunction has occurred that could lead to an incorrect position value or require user intervention | See Semantics Section |
| 45 | Conditional | Get | NV | Supported Alarms | WORD | Information about supported *Alarms* | See Semantics Section |
| 46 | Conditional | Get | V | Alarm Fag | BOOL | Indicates that an alarm error occurred. | See Semantics Section 0 = OK 1 = Alarm error |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 47 | Optional | Get | V | Warnings | WORD | Internal parameters exceeded | See semantics section. |
| 48 | Conditional | Get | NV | Supported Warnings | WORD | Information about supported *Warnings* | See Semantics Section |
| 49 | Conditional | Get | V | Warning Flag | BOOL | Indicates that a warning error occurred | See Semantics Section<br>0 = OK<br>1 = Warning flag |
| 50 | Optional | Get | NV | Operating Time | UDINT | Stores operating time for the encoder in tenths of an hour | |
| 51 | Conditional | Get | NV | Offset Value | DINT | The Offset value is calculated by the preset function. Shift position value with the calculated value | See Semantics Section |

Table Footnotes:

1 One and only one of the attributes *Position Value Signed* or *Position Value Unsigned* shall be implemented depending on the application requirements.

## 5A-23.4 Semantics

### 5A-23.4.1 Position Value Unsigned - Attribute 3

This attribute represents the absolute position detected by the position sensor conditioned by the *Value Bit Resolution* and *Zero Offset* attributes. Refer to the following descriptions of the *Value Bit Resolution* and *Zero Offset* attributes for details.

### 5A-23.4.2 Position Value Signed - Attribute 10

This attribute represents the absolute position detected by the position sensor. It is not conditioned by the *Value Bit Resolution* and *Zero Offset* attributes.

### 5A-23.4.3 Value Bit Resolution – Attribute 5

This attribute specifies the number of significant bits used for the *Position Value Unsigned* (attribute 3). The raw value is shifted left or right to supply the indicated number of significant bits.

**Table 5A-23.4 Bit Resolution Attribute Value**

| Resolution | Value = |
|---|---|
| > Physical Resolution | (RawValue << (PhysicalResolution - Bit Resolution)) + ZeroOffset |
| < Physical Resolution | (RawValue >> (Bit Resolution - PhysicalResolution)) + ZeroOffset |
| = Physical Resolution | RawValue + ZeroOffset |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-23.5 Example Bit Resolution Values**

| Raw Value | Resolution | Adjusted Val. | Notes |
|---|---|---|---|
| 10 bit (0 to $3FF_{hex}$) | 8 | 0 to $FF_{hex}$ | Bit Resolution < Physical Resolution, surplus bits are discarded |
| 6 bit (0 to $3F_{hex}$) | 8 | 0 to $FF_{hex}$ | Bit Resolution > Physical Resolution, missing bits are zero. Adjusted value will actually be 0 to $FC_{hex}$ in multiples of 4 |
| 10 bit (0 to $3FF_{hex}$) | 10 | 0 to $3FF_{hex}$ | Bit Resolution = Physical Resolution, no conversion. |

### 5A-23.4.4 Zero Offset – Attribute 6

The Zero Offset Attribute adjusts the zero point of Value. Zero Offset is added to Value to adjust the zero point. The Zero Offset Attribute is applied *after* the *Value Bit Resolution* Attribute.

If the result of the addition exceeds the maximum specified by the Resolution attribute the overflow bits are discarded.

**Table 5A-23.6 Example Zero Offset Values**

| Adjusted Val. | Zero Offset | Resolution | Value |
|---|---|---|---|
| 0 | 10 | 8 | 10 |
| 250 | 0 | 8 | 250 |
| 250 | 20 | 8 | 15 [1] |
| 250 | 20 | 10 | 270 |
| 250 | 255 | 8 | 249 [2] |

Table Footnotes:
1. Value overflowed
2. Value underflowed

#### 5A-23.4.4.1 CAM, CAM Low Limit, CAM High Limit – Attributes 4, 7 and 8

The *CAM* attribute is a virtual CAM switch. The state of the *CAM* attribute is determined by the *CAM Low Limit*, *CAM High Limit* and *Position Value Unsigned* attributes. The *Position Value Unsigned* attribute is used after the *Value Bit Resolution* and *Zero Offset* attributes have been applied.

**Table 5A-23.7 CAM, CAM Low Limit & CAM High Limit Operation**

| CAM Low Limit | CAM is On (true, 1) if ... | CAM is Off (false, 0) if ... |
|---|---|---|
| > CAM High Limit | Value > CAM Low *or* Value < CAM High | Value < CAM Low *and* Value > CAM High |
| < CAM High Limit | Value > CAM Low *and* Value < CAM High | Value < CAM Low *or* Value > CAM High |
| = CAM High Limit | Never | Always |

### 5A-23.4.5 Auto Zero – Attribute 9

This attribute controls the auto-zero feature of the resolver. A rising edge (transition from 0 to 1) on this attribute adjusts the *Zero Offset* attribute to a value that results in the *Position Value Unsigned* attribute being zero.

If the *Zero Offset* attribute is implemented as non-volatile, the AutoZero command must store the new *Zero Offset* value.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-23.4.6 Position Sensor Type – Attribute 11

**Table 5A-23.8 Attribute 11, Position Sensor Type Values**

| Value | Definition |
|---|---|
| 00 | Single Turn resolver (value if attribute is not supported) |
| 01 | Single-Turn absolute rotary encoder |
| 02 | Multi-Turn absolute rotary encoder |
| 03 | Single-Turn absolute rotary encoder with electronic turn count |
| 04 | Incremental rotary encoder |
| 05 | Incremental rotary encoder with electronic counting |
| 06 | Incremental linear encoder |
| 07 | Incremental linear encoder with electronic counting |
| 08 | Absolute linear encoder |
| 09 | Absolute linear encoder with cyclic coding |
| 10 | Multi-Sensor encoder interface |
| 11 | Multi-Turn absolute rotary encoder with electronic turn count |
| 12 | Virtual Axis Sensor |
| 13... 65535 | Reserved by CIP |

### 5A-23.4.7 Direction Counting Toggle – Attribute 12

For linear devices, the *Position Value* attribute increases when the value of the *Direction Counting Toggle* attribute is FORWARD/CW (0). If the value of the *Direction Counting Toggle* attribute is REVERSE/CCW (1), the *Position Value* attribute decreases. The *Direction Counting Toggle* defines the increasing *Position Value* 1) for rotary devices as clockwise shaft rotation or counter clockwise as viewed facing the mounting face (CW = 0, CCW = 1), 2) for linear devices as Forward, moving away from the electrical connection, or Reverse, moving toward the electrical connection, as viewed from the mounting face. Changing this value shall change the sign of velocity and position relative to physical movement.

For rotary encoders the code sequence defines whether increasing or decreasing position values are output when the encoder shaft rotates clockwise (CW) or counterclockwise (CCW) as seen on the shaft. Example: By turning the shaft clockwise the *Position Value* attribute will increase when *Direction Counting Toggle* is defined as clockwise (0=CW).

### 5A-23.4.8 Commissioning Diagnostic Control – Attribute 13

When the *Commissioning Diagnostic Control* attribute is set to ON (1) it is possible to check the encoder components responsible for position detection at encoder stand still. This enables an extensive check of the correctness of the position values.

If errors are detected, they will be indicated by the respective bits in the *Alarm* attribute. If this attribute is set to OFF (0), no diagnostics will be executed.

### 5A-23.4.9 Scaling Function Control – Attribute 14

When the *Scaling Function Control* attribute is set to ON (1), the *Position Value* attribute is converted from the physical resolution of the device to position units.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

If this attribute is implemented and turned OFF, all functions within position sensor shall continue to use the scaled value, except the *Position Value* (attribute 10) reported shall be the raw, unscaled value. This attribute is provided solely for calibration and troubleshooting purposes and does not affect scaling when *Position Format* (attribute 15) and *Measuring Units per Span* (attribute 16) are implemented.

The Measuring Units per Revolution and Total Measuring Range in Measuring Units attributes are the scaling parameters.

For rotary devices:

If Scaling Function Control = = OFF

Then Position Value = physical resolution of device in counts

If Scaling Function Control = = ON

The Position Value = physical resolution of device in counts * (Measuring Units per Span (attribute 16) / Physical Resolution Span (attribute 42))

If *Measuring Units per Span* and *Total Measuring Range in Measuring Units* are not supported, the scaling of the *Position Value* is not provided and the *Position Value* is always equal to physical resolution of the device in counts.

If *Scaling Function Control* is not supported, the *Position Value* shall always be scaled based upon the values within *Position Format* and *Measuring Units per Span* attributes. When the *Scaling Function Control* attribute is supported and is enabled (ON=1), P*osition Value* is calculated according to the formula above. When disabled, physical resolution shall be contained in *Position Value*.

### 5A-23.4.10 Position Format – Attribute 15

This attribute identifies the engineering units for the *Position Value* attribute (attribute 3 or 10). *Position Format* is a component of all other attributes containing a distance data type, like *Velocity Value* (attribute 24) and *Acceleration Value* (attribute 29). Engineering units like counts, mm or inches are allowed.

The *Position Format* (attribute 3 or 10) identifies the engineering units for one increment of the *Position Value*. The *Position Measuring Increment* (attribute 18) indicates the smallest granularity that the *Position Value* (attribute 3 or 10) shall change.

### 5A-23.4.11 Measuring Units per Span – Attribute 16

The *Measuring Units per Span* sets the number of distinguishable (desired) steps per unit of travel. Rotary units would contain the counts per one complete span.

### 5A-23.4.12 Total Measuring Range in Measuring Units – Attribute 17

The parameter *Total Measuring Range in Measuring Units* sets the number of distinguishable steps over the total measuring range. This value must be less than maximum physical resolution of the device. Maximum physical resolution should be listed on the type plate.

This parameter is used for rotary and linear devices..

### 5A-23.4.13 Position measuring increment – Attribute 18

This attribute may be supported when *Position Format* (attribute 15) can be set to a value other than count (0x1001). The *Position Measuring Step* attribute defines the measuring step settings for the position for linear encoders. Basic position measuring step in 0.001 μm or 0.1nanoinch is affected by *Position Format* (attribute 15).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-23.4.14 Preset Value – Attribute 19

This attribute supports adapting  a desired position value to an actual position value.

At the instant a "Set Attribute" to attribute #19 is performed, the following occurs:

Attribute #19 (*Preset Value*) is set to the service data of the request.

*Offset Value* (Attribute #51) is set to the value resulting from [*Preset Value*] - [*Position Value*]

During operation, the formula is immaterial and the behavior of the device is:

*Position Value* = (internal position value) + *Offset Value*

The value contained in the *Preset Value* attribute is meaningless (with respect to the indicated position) after the Set Attribute has taken effect.  The *Offset Value* remains a constant and the now reported *Position Value* is the adjusted position value using the offset applied to some (internal) raw position value.

The non-settable attribute (from explicit messaging) is actually set (e.g. the *Offset Value* attribute must change) by an internal mechanism.

*Preset Value* (attribute 19) = *Position Value* (attribute 10) + *Offset Value* (attribute 51).

### 5A-23.4.15 COS Delta – Attribute 20

A COS I/O message will be generated when the *Position Value* (attribute 3 or 10) changes by this value. Setting this value to 0 disables COS delta limit for *Position Value* (attribute 3 or 10) changes and all position changes shall generate a COS message.

### 5A-23.4.16 Position State Register, Position limits – Attributes 21 to 23

The *Position Low Limit* and *Position High Limit* attributes configure the actual work area. The *Position State Register* contains the actual area status of the encoder position. If the position is out of range, a bit will be set in the *Position State Register* attribute. If the position is lower than the position value set in *Position Low Limit*, then bit 2 flags the underflow. If the position is higher than the position value set in *Position High Limit*, then bit 1 flags the overflow. The *Position Limits* define a configurable work area within the measuring range. This function allows a replacement of external proximity switches.

**Table 5A-23.9 Position State Register Structure**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Reserved | Reserved | Reserved | Reserved | Reserved | Range underflow | Range overflow | Out of range |

### 5A-23.4.17 Velocity Format– Attribute 25

This *Velocity Format* attribute identifies the engineering units for the *Velocity Value* attribute (attribute 24). Default is counts (steps) per second.  Any reasonable format from Appendix D-2.30 may be used depending upon application requirements. *Velocity Format* is a component of all other attributes containing velocity for calculation like acceleration. If this attribute is changed, both *Minimum Velocity* and *Maximum Velocity* Setpoints shall be reset to their default values. This attribute is required when *Velocity Value* is implemented. Otherwise, this attribute is not implemented.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-23.4.18 Velocity Resolution – Attribute 26

The parameter *Velocity Resolution* defines the resolution of the *Velocity Value* attribute. Basic velocity resolution is defined in steps of 0.01 mm /s or 0.001 inch/s, affected by the chosen *Position Format* (attribute 15). This attribute is required when *Velocity Value* is implemented. Otherwise, this attribute is not implemented.

### 5A-23.4.19 Minimum Velocity / Maximum Velocity - Attribute 27 thru 28

The actual velocity speed limit values with minimum and maximum can be configured in attributes 27 and 28. Corresponding flags in *Status Warnings* (attribute 68) are affected. These attributes are optional when *Velocity Value* is implemented.

### 5A-23.4.20 Acceleration Value - Attribute 29

Acceleration is defined as the change of velocity (speed) per time unit. The time unit used should be seconds. This means the *Acceleration Value* is given in "*Velocity Format*" per second. The current *Acceleration Value* is derived from the calculated *Velocity Value*. The value is signed with the following meaning.

**Table 5A-23.10 Acceleration Value Attribute**

| Value (sign) | Explanation |
|---|---|
| 0 | Velocity is constant |
| + | Velocity is increasing |
| - | Velocity is decreasing |

### 5A-23.4.21 Acceleration Format – Attribute 30

This attribute identifies the engineering units for the *Acceleration Value* (attribute 29). Default is meters/second. Any reasonable format from Appendix D-2.8 may be used depending upon application requirements. If this attribute is changed, both *Minimum Acceleration Setpoint* and *Maximum Acceleration Setpoint* shall be reset to their default values. This attribute is required when *Acceleration Value* is implemented. Otherwise, this attribute is not implemented.

### 5A-23.4.22 Acceleration Resolution – Attribute 31

The parameter *Acceleration Resolution* defines the resolution of the *Acceleration Value* attribute. Basic *Acceleration Resolution* is defined in steps of 1 mm/s$^2$ or 0,1 inch/s$^2$, affected by the chosen *Position Format* [attribute15]. This attribute is required when *Acceleration Value* is implemented. Otherwise, this attribute is not implemented.

### 5A-23.4.23 Minimum Acceleration / Maximum Acceleration - Attribute 32 thru 33

The actual acceleration limit values with minimum and maximum can be configured in the attributes 32 and 33. Corresponding flags in status *Warnings* (attribute 47) are affected. These attributes are optional when *Acceleration Value* is implemented. Otherwise, these attributes are not implemented.

### 5A-23.4.24 Number of CAM Channels – Attribute 34

This attribute defines the number of cam channels that are supported by an encoder device. Each cam has parameters for the minimum switch point, the maximum switch point and setting a hysteresis to the switch points. If this attribute is implemented then attributes 35-40 shall be implemented. If this attribute is NOT implemented then attributes 35-40 shall NOT be implemented.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5A-23.1 Possiblle Usage of Cam's and Switch Points**

**Figure 5A-23.2 Possibe Usage of Hysteresis**

A Low Limit trip occurs at the lower "A" value (dashed left down arrow) and is rest at "B" threshold (the higher solid left up arrow). A High Limit trip occurs at the higher "D" value (dashed right down arrow) and is reset at the lower "C" threshold (solid right up arrow).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5A-23.3 Principle of CAM Channels**

| | | |
|---|---|---|
| CAM Channel state register 1 | CAM Channel state register 2 | CAM Channel state register N |
| CAM Channel polarity register 1 | CAM Channel polarity register 2 | CAM Channel polarity register N |
| CAM Channel enable register 1 | CAM Channel enable register 2 | CAM Channel enable register N |
| CAM Channel hysteresis register 1 | CAM Channel hysteresis register 2 | CAM Ch. hysteresis register N |

Where: N = Number of CAM channels

## 5A-23.4.25 CAM Channel State Register – Attribute 35

This attribute defines the status bit of the cam. The status bit set to 1 defines “cam active”. The status bit set to 0 defines “cam inactive”. If the *CAM Polarity Register* attribute of a cam is set to one, the actual cam state will be inverted. The *CAM Channel State Register* array size is determined by the value of the *Number of CAM Channels* attribute 34.

**Table 5A-23.11 Cam Channel State register for CAM Channel N**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| CAM_ 8 | CAM_ 7 | CAM_ 6 | CAM_ 5 | CAM_ 4 | CAM_ 3 | CAM_ 2 | CAM_ 1 |
| State | State | State | State | State | State | State | State |

N = Number of CAM channel, with respect to Attribute 34

## 5A-23.4.26 CAM Channel Polarity Register – Attribute 36

This attribute contains the actual polarity settings for *Cam Channel State Register*. If the *CAM Polarity Register* bit is set to 1, the cam state of an active cam will signal by setting the related cam state bit to zero. In the other case, the cam state of the related cam will not be inverted.

**Table 5A-23.12 CAM Polarity Register of CAM Channel N**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| CAM_8 | CAM_7 | CAM_6 | CAM_5 | CAM_4 | CAM_3 | CAM_2 | CAM_1 |
| Polarity | Polarity | Polarity | Polarity | Polarity | Polarity | Polarity | Polarity |

N = Number of CAM channel, with respect to Attribute 34

## 5A-23.4.27 CAM Channel Enable Register – Attribute 37

The *Cam Channel Enable Register* contains the calculation state for the respective cams. If the *Cam Channel Enable Register* bit is set to 1, the cam state will be calculated by the device. In the other case the cam state of the related cam will be set permanently to 0.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-23.13 CAM Enable Register of CAM Channel N**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| CAM_8 | CAM_7 | CAM_6 | CAM_5 | CAM_4 | CAM_3 | CAM_2 | CAM_1 |
| Enable | Enable | Enable | Enable | Enable | Enable | Enable | Enable |
| N=Number of CAM channel, with respect to Attribute 34 | | | | | | | |

### 5A-23.4.28 CAM Low Limit – Attribute 38

Each cam channel contains the switch point for the lower limit setting for a maximum of 8 cams for one cam channel. The *CAM Low Limit* array size is determined by the value of the *Number of CAM Channels* attribute 34. The default value for this attribute is zero.

### 5A-23.4.29 CAM High Limit – Attribute 39

Each cam channel contains the switch point for the higher limit setting for a maximum of 8 cams for one cam channel. The *CAM High Limit* array size is determined by the value of the *Number of CAM Channels* attribute 34. The default value for this attribute is zero.

### 5A-23.4.30 CAM Hysteresis – Attribute 40

The *CAM Hysteresis* value is added to the *CAM High Limit* and subtracted from the *CAM Low Limit* when calculating the CAM state. The *CAM Hysteresis* array size is determined by the value of the *Number of CAM Channels* attribute 34.

### 5A-23.4.31 Operating Status – Attribute 41

This attribute contains the operating status of the encoder.

**Table 5A-23.14 Operating Status Attribute Bit Definitions**

| Bit | Description | FALSE (0) | TRUE (1) |
|---|---|---|---|
| 0 | Direction | Increasing | Decreasing |
| 1 | Scaling | Off | On |
| 2..4 | Reserved by CIP | | |
| 5..7 | Vendor specific | | |

### 5A-23.4.32 Physical Resolution – Attribute 42

This is the physical resolution of the position sensor. For rotary devices, the number of steps per span can be read out. For linear devices, the units are steps per linear unit (nanometer or 0.1 nanoinch) the measuring step is given for linear encoder.

### 5A-23.4.33 Number of Spans – Attribute 43

For a multi-Turn device the *Number of Spans* and the *Physical Resolution* (attribute 42) gives the physical measuring range to the formula below.

Physical Measuring range = Physical Resolution * Number of Spans


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-23.4.34 Alarms – Attribute 44

An alarm is set if a malfunction bit is set to true (high). The alarm remains active until the alarm is cleared and the device is able to provide an accurate position value.

**Table 5A-23.15 Alarms Attribute Bit Definitions**

| Bit | Description | FALSE (0) | TRUE (1) |
|---|---|---|---|
| 0 | Position error | NO | YES |
| 1 | Diagnostic error | NO | YES |
| 2…11 | Reserved by CIP | | |
| 12…15 | Vendor specific | | |

### 5A-23.4.35 Supported Alarms – Attribute 45

This attribute contains information on supported alarms by the position sensor device. This attribute is required when the *Alarms* attribute is implemented.

**Table 5A-23.16 Supported Alarms Attribute Bit Definitions**

| Bit | Description | FALSE (0) | TRUE (1) |
|---|---|---|---|
| 0 | Position error | Not supported | Supported |
| 1 | Diagnostic error | Not supported | Supported |
| 2…11 | Reserved by CIP | | |
| 12…15 | Vendor specific | | |

### 5A-23.4.36 Alarm Flag – Attribute 46

Indicates that an alarm error has occurred. This attribute is the logical OR of all the alarm bits in the *Alarms* attribute (attribute 44). This attribute is required when the *Alarms* attribute is implemented.

### 5A-23.4.37 Warnings – Attribute 47

The *Warnings* attribute indicates that tolerance for certain internal parameters of the device have been exceeded. In contrast to alarms, warnings do not imply incorrect position values. All warnings are cleared if the tolerances are again within normal parameters. For the operating time limit warning (bit 3) the warning is only set again after a power-on sequence. The *Warning Flag* attribute indicates if any of the defined warnings are active.

**Table 5A-23.17 Warnings Attribute Bit Definitions**

| Bit | Description | FALSE (0) | TRUE (1) |
|---|---|---|---|
| 0 | Frequency Exceeded | NO | YES |
| 1 | Light Control reserve | Not reached | Error |
| 2 | CPU Watchdog | OK | Reset generated |
| 3 | Operating Time Limit Warning | NO | YES |
| 4 | Battery charge | OK | Too low |
| 5 | Reference Point | Reached | Not reached |
| 6 | Minimum Velocity Flag | OK | Fall below |
| 7 | Maximum Velocity Flag | OK | Exceeded |
| 8 | Minimum Acceleration Flag | OK | Fall below |
| 9 | Maximum Acceleration Flag | OK | Exceeded |
| 10 | Position Limits Exceeded | OK | Exceeded |
| 11-12 | Reserved by CIP | Always 0 | |
| 13-15 | Vendor specific | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-23.4.38 Supported Warnings – Attribute 48

This attribute contains information on supported warnings by the position sensor device. This attribute is required when the *Warnings* attribute is implemented.

**Table 5A-23.18 Supported Warnings Attribute Bit Definitions**

| Bit | Description | FALSE (0) | TRUE (1) |
|---|---|---|---|
| 0 | Frequency Exceeded | Not supported | Supported |
| 1 | Light Control reserve | Not supported | Supported |
| 2 | CPU Watchdog | Not supported | Supported |
| 3 | Operating Time Limit Warning | Not supported | Supported |
| 4 | Battery charge | Not supported | Supported |
| 5 | Reference Point | Not supported | Supported |
| 6 | Minimum Velocity Flag | Not supported | Supported |
| 7 | Maximum Velocity Flag | Not supported | Supported |
| 8 | Minimum Acceleration Flag | Not supported | Supported |
| 9 | Maximum Acceleration Flag | Not supported | Supported |
| 10 | Position Limits Exceeded | OK | Exceeded |
| 11-12 | Reserved by CIP | Always 0 | |
| 13-15 | Vendor specific | | |

### 5A-23.4.39 Warning flag – Attribute 49

Indicates that warning error has occurred. This attribute is the logical OR of all the warnings bits in the *Warnings* attribute (attribute 47). This attribute is required when the *Warnings* attribute is implemented.

### 5A-23.4.40 Operating Time – Attribute 50

This attribute is incremented as long as the encoder is powered. The *Operating Time* value is presented in tenths (0.1) of an hour.

### 5A-23.4.41 Offset Value – Attribute 51

This attribute is required when the *Preset Value* (attribute19) is implemented.

The *Offset Value* attribute is calculated by the preset function and shifts the *Position Value* attribute with the calculated value. The *Offset Value* is stored automatically by the device and can be read from the encoder for diagnostic purposes.

*Offset Value* (attribute 49) = *Preset Value* (attribute 19) - *Position Value* (attribute10).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-23.5 Common Services

The Position Sensor Object provides the following Common Services:

**Table 5A-23.19 Position Sensor Object Common Service**

| Service Code | Need In Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0x05 | Optional | NA | Reset | Resets all parameter values to the factory default |
| 0x0D | Optional | NA | Apply_Attributes | Cause the configuration to become active |
| 0x0E | Conditional [1] | Required | Get_Attribute_Single | Returns the contents of the specified attribute |
| 0x10 | N/A | Optional | Set_Attribute_Single | Modifies an attribute value |
| 0x15 | Optional | NA | Restore | Restores all parameter values from non-volatile storage |
| 0x16 | Optional | NA | Save | Saves all parameters to non-volatile storage |
| 0x18 | N/A | Conditional [2] | Get_Member | Returns an element of an array for specified attribute. Specifically intended for usage on “CAM” attributes. |
| 0x19 | N/A | Conditional [2] | Set_Member | Modifies an element of an array for specified attribute. Specifically intended for usage on “CAM” attributes. |

Table Footnotes:

1 The Get_Attribute_Single service is *required* at the class level if any class attributes are implemented

2 The Get_Member and Set_Member services are required when any of the array attributes are implemented.

See Appendix A for definition of these services

If several parameters will be modified resulting in side effects the Apply_Attributes service shall be implemented to cause the configuration to become active after all parameters have been set. The Apply_Attributes service shall validate parameter settings. If any parameters are in conflict, the error response Invalid value (0x09) shall be returned with the extended error code indicating the conflicting attribute Id.

If Save or Restore service is implemented, all implemented attributes of the Position Sensor Object must be restored or saved to non-volatile storage.

The reset service has the following parameter:

**Table 5A-23.20 Reset Service**

| Name | Type | Description of Request Parameters | Semantics of Values |
|---|---|---|---|
| Type | USINT | Type of Reset | See Table below |

The parameter Type for the Reset service has the following bit specifications:

**Table 5A-23.21 Reset Service Parameter Values**

| Value | Type of Reset |
|---|---|
| 0 | Emulate as closely as possible cycling power. This value is the default if this parameter is omitted. |
| 1 | Return as closely as possible to the out-of-box configuration, then emulate cycling power as closely as possible. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-23.6 Object–specific Services

The Position Sensor Object provides no Object–specific services.

## 5A-23.7 Behavior

The State Transition Diagram (Figure 5A-23.4) provides a graphical description of the events and corresponding state transitions. A subset of the states and events may be supported in an application, but the behavior must be consistent.

The State Event Matrix (See Table 5A-23.23) lists all pertinent events and the corresponding action to be taken while in each state.

**Important:** Events can occur simultaneously, but the *Fault* events have priority if they occur simultaneously with other events.

**Figure 5A-23.4 State Transition Diagram for Position Sensor Object**

LED = I/O Status LED

**Important:** Events can occur simultaneously, but the *Fault* events have priority if they occur simultaneously with other events.
The following SEM contains these states:

- ***Non–Existent:*** a module without power.
- ***Available:*** waiting for a connection, power–up discrete input point defaults are set.
- ***Run:*** Position Sensor sensing data from its input and transmitting the data.
- ***Recoverable Fault:*** a recoverable fault has occurred.
- ***Unrecoverable Fault:*** an unrecoverable fault has occurred.

The SEM also contains these events:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-23.22 Position Sensor Object Event Definitions**

| This event | Is |
|---|---|
| Sample Trigger | a change of state, cyclic timer trigger, application trigger |
| Connection Deleted | I/O connection deleted. |
| Apply_Attributes | the Apply service of the I/O connection object the Position Sensor Object is connected to. Note: the application is responsible for validating the connection object's attributes. |
| Fault Cleared | the application clearing a detected fault |
| Connection Transitions to Established | I/O connection transitions to Established. |
| Connection Transitions to Timed Out state | I/O connection transitions to Timed-Out |

The figure below is a **conceptual** illustration of the state machine for a typical input object (producing application). The events listed above are represented by the dotted line labeled "state change."

**Figure 5A-23.5 Conceptual Position Sensor Object Operation**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-23.23 State Event Matrix for the Position Sensor Object**

| Event | State | | | | |
|---|---|---|---|---|---|
| | **Non-Existent** | **Available** | **Run** | **Recoverable Fault** | **Unrecoverable Fault** |
| Sample Trigger | Not Applicable | Ignore event | Sample data, Send data | Ignore event | Ignore event |
| Apply Attributes | Not Applicable | Verify attributes, return result | Return error (Object State Conflict) | Return error (Object State Conflict) | Ignore event |
| Connection Deleted | Not Applicable | Ignore Event | Transition to Available | Transition to Available | Ignore event |
| Connection Transitions to Established | Not Applicable | Transition to Run | Ignore event | Ignore event | Ignore event |
| Connection Transitions to Timed Out state | Not Applicable | Ignore event | Transition to Recoverable Fault | Ignore event | Ignore event |
| Fault Cleared | Not Applicable | Not Applicable | Not Applicable | Transition to Run | Ignore event |
| Get_Attribute | Return Error (Object Does Not Exist) | Return value | Return value | Return value | Return Value |
| Set_Attribute | Return Error (Object Does Not Exist) | Accept value | Accept value | Accept value | Ignore event |
| I/O Status LED | Off | Off | Solid Green | Flash Red | Solid Red |
| Reset | Not Applicable | Perform service | Perform service | Attempt to Recover | Ignore Event |
| Restore | Not Applicable | Restore from storage | Restore from storage | Ignore Event | Ignore Event |
| Save | Not Applicable | Stores parameter to non-volatile memory | Stores parameter to non-volatile memory | Ignore Event | Ignore Event |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-24 Position Controller Supervisor Object

**Class Code: 24 hex**

The position controller supervisor handles errors for the position controller as well as Home and Registration inputs.

## 5A-24.1 Revision History

Since the initial release of this object class definition, changes have been made that require a revision update of this object class. The table below represents the revision history:

**Table 5A-24.1 Position Controller Supervisor Object Revision History**

| Revision | Description |
|---|---|
| 01 | Initial release |
| 02 | Class Attributes 32 and 33 added |

## 5A-24.2 Class Attributes

**Table 5A-24.2 Position Controller Supervisor Object Class Attributes**

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Get | Revision | UINT | Revision of this object. | The current value assigned to this attribute is two (02). |
| 2 thru 7 | These class attributes are either optional or conditional and are described in Chapter 4 of this specification. | | | | | |
| 32 | Required | Get | Consumed Axis Selection Number | USINT | Specifies the axis number to which the data contained in the I/O Command Message is routed. | Value in the range of 1-7 |
| 33 | Required | Get | Produced Axis Selection Number | USINT | Specifies the axis number to which the data contained in the I/O Response Message is routed. | Value in the range of 1-7 |

### 5A-24.2.1 Consumed Axis Selection Number

When an I/O Message is consumed by the Position Controller Supervisor object, its internal destination may vary. The destination of the I/O Message shall be specified within the *Command Axis Number*. See the description of I/O Connection Messages of the Position Controller Device Profile in Chapter 6 for details.

For static systems, this attribute is normally specified in the *Consumed Connection Path* of the respective I/O Connection instance.

### 5A-24.2.2 Produced Axis Selection Number

When an I/O Message is produced by the Position Controller Supervisor object, its internal source may vary. The source of the I/O Message shall be specified within the *Response Axis Number*. See the description of I/O Connection Messages of the Position Controller Device Profile in Chapter 6 for details.

For static systems, this attribute is normally specified in the *Produced Connection Path* of the respective I/O Connection instance.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-24.3 Object Instance Attributes

### 5A-24.3.1 Supervisor Attributes

**Table 5A-24.3 Position Controller Supervisor Instance Attributes – Supervisor Attributes**

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Optional | Get | Number of Attributes | USINT | The total number of attributes supported by this object in this device | Return value is in the range of 0 to 255. |
| 2 | Optional | Get | Attribute List | Array of USINT | Returns an array with a list of the attributes supported by this object in this device. | Array size defined by attribute 1. |
| 3 | Required | Get | Axis Number | USINT | Returns the axis number which is the same as the instance for this object. | This value will be in the range of 1 to 7.<br>The axis number is the same as the instance number for all of the axis objects: position controller supervisor, position controller, drive, motor data, and block sequencer. |
| 4 | | | Reserved | | | |
| 5 | Required | Get | General Fault | BOOL | General Fault flag. This bit is logical OR of all fault condition attribute flags in the device. This bit is reset when the fault condition is removed. | 1 = fault condition exists. |
| 6 | Required | Set | Command Message Type | USINT | Sets the command message type that is being sent by the controlling device. | Valid Message Type codes are 1 to 1F hex.<br>1 = Type 01 hex<br>2 = Type 02 hex, etc. |
| 7 | Required | Set | Response Message Type | USINT | Sets the response message that is returned to the controlling device | Valid Message Type codes are 1 to 1F hex..<br>1 = Type 01 hex<br>2 = Type 02 hex etc. |
| 8 | Optional | Get | Fault Input | BOOL | Fault input fault | 1 = fault input is active. |
| 9 | Optional | Set | Fault Input Action | USINT | Action taken when fault input becomes active code. | 0 = Command Output Generator Off,<br>1 = Hard Stop,<br>2 = smooth stop,<br>3 = no action.<br>4 - 127 = Reserved by CIP<br>128 - 255 = Vendor specific |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-24.3.2 Home and Index Attributes

**Table 5A-24.4 Position Controller Supervisor Instance Attributes – Home and Index Attributes**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 10 | Optional | Set | Home Action | USINT | Home input action code. Action taken when armed home input is triggered. | 0 = Command Output Generator off,<br>1 = Hard stop,<br>2 = Smooth stop,<br>3 = No action,<br>4 = Gate index.<br>5 - 127 = Reserved by CIP<br>128 - 255 = Vendor specific |
| 11 | Optional | Set | Home Active level | BOOL | Home trigger Active Level flag is used to program the Home inputs active level. | 0 = active low, 1 = active high. |
| 12 | Optional | Get/Set | Home Arm | BOOL | Home trigger arm flag is used to arm the Home input. | 1 arms the home input, reading a 0 indicates the trigger has occurred. |
| 13 | Optional | Set | Index Action | USINT | Index input action code. | 0 = Command Output Generator off,<br>1 = Hard stop and,<br>2 = Smooth stop,<br>3 = No action.<br>4 - 127 = Reserved by CIP<br>128 - 255 = Vendor specific |
| 14 | Optional | Get/Set | Index Active Level | BOOL | Used to program the Index inputs active level. | 0 = active low, 1 = active high. |
| 15 | Optional | Get/Set | Index Arm | BOOL | Index trigger arm flag is used to arm the Index input. | 1 arms the index input, reading 0 indicates the trigger has occurred. |
| 16 | Optional | Get | Home Input Level | BOOL | Actual level of the Home input. | 0 = Home input is low<br>1 = Home input is high. |
| 17 | Optional | Get | Home Position | DINT | Home trigger position reflects the position at the time the home input is triggered. | This value can be in the range of 0x80000001 to 0x7FFFFFFF. |
| 18 | Optional | Get | Index Position | DINT | Index trigger position reflects the position at the time the home input is triggered. | This value can be in the range of 0x80000001 to 0x7FFFFFFF. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-24.3.3 Registration Attributes

**Table 5A-24.5 Position Controller Supervisor Instance Attributes – Registration Attributes**

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 19 | Optional | Set | Registration Action | USINT | Registration input action defines what happens when the registration input is triggered. | 0 = Command Output Generator off<br>1= Hard Stop<br>2 = Smooth stop<br>3 = No action.<br>4 = Go to Reg position offset<br>5 = Go to Reg position absolute.<br>6 - 127 = Reserved by CIP<br>128 - 255 = Vendor specific |
| 20 | Optional | Set | Registration Active level | BOOL | Registration trigger Active Level flag is used to program the Registration inputs active level. | 0 = active low, 1 = active high. |
| 21 | Optional | Get/Set | Registration Arm | BOOL | Registration trigger arm flag is used to arm the Registration input. | Set to 1 to arm the registration input, reading a 0 indicates the registration trigger has occurred. |
| 22 | Optional | Get | Registration Input Level | BOOL | Actual level of the registration input. | 0 = registration is low<br>1 = registration is high. |
| 23 | Optional | Set | Registration Offset | DINT | Defines a position value that is used as an offset or absolute position dependent on the registration action code. | This value can be in the range of 0x80000001 to 0x7FFFFFFF. |
| 24 | Optional | Get | Registration Position | DINT | Registration trigger position reflects the position at the time the Registration input is triggered. | This value can be in the range of 0x80000001 to 0x7FFFFFFF. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-24.3.4 Axis Following Attributes

**Table 5A-24.6 Position Controller Supervisor Instance Attributes – Axis Following Attributes**

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 25 | Optional | Set | Follow Enable | BOOL | Follow Enable enables following of the Follow Axis. | 0 = following disabled,<br>1 = following enabled. |
| 26 | Optional | Set | Follow Axis | USINT | Specifies the Axis to follow. | 0 = no following,<br>1 to 255 specifies the axis. |
| 27 | Optional | Set | Follow Divisor | DINT | Used to calculate the Command Position by dividing the Follow Axis position with this value. | |
| 28 | Optional | Set | Follow Multiplier | DINT | Used to calculate the Command Position by multiplying the Follow Axis position with this value. | |

## 5A-24.4 Common Services

**Table 5A-24.7 Position Controller Supervisor Common Services**

| Need in Implementation | Name | Service Code | Description of Service |
|---|---|---|---|
| Required | Get_Attribute_Single | 0Ehex | Returns the contents of the specified attribute |
| Required | Set_Attribute_Single | 10hex | Modifies the attribute value. |
| See Appendix A for definitions of these services | | | |

## 5A-24.5 Object-specific Services

The Position Controller Supervisor object provides no object-specific services.

## 5A-24.6 Behavior

Object behavior is defined by the following state diagrams.

### 5A-24.6.1 Position Controller Supervisor State Diagrams

#### 5A-24.6.1.1 Home Input State Diagrams

The following state diagram describes the behavior of the Home input. The Home input active level can be programmed. Home will trigger when it is armed and the input goes to the active level. When home is triggered the home action is performed and the home input returns to the disabled state.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5A-24.1 Home Input State Diagram**

### 5A-24.6.1.2 Index Input State Diagram

The following diagram describes the behavior of the Index input. The Index input active level can be programmed. Index will trigger when it is armed and the input goes to the active level. In addition the index input can be gated using the Home input when the Home Action attribute is set to Gate index. When index is triggered the index action is performed and the Index input returns to the disabled state.

**Figure 5A-24.2 Index Input State Diagram**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-24.6.1.3 Registration Input State Diagram

The following state diagram describes the behavior of the Registration input. The Registration input active level can be programmed. The Registration input will trigger when it is armed and the input goes to the active level. When the Registration input is triggered the Registration action is performed and the Registration input returns to the disabled state.

**Figure 5A-24.3 Registration Input State Diagram**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-25 Position Controller Object

## Class Code: 25 hex

The position controller object performs the control output velocity profiling and handles input and output to and from the motor drive unit, limit switches registration etc.

## 5A-25.1 Revision History

Since the initial release of this object class definition, changes have been made that require a revision update of this object class. The table below represents the revision history:

**Table 5A-25.1 Position Controller Object Revision History**

| Revision | Description |
|---|---|
| 01 | Initial release |
| 02 | Instance Attribute 58 added |

## 5A-25.2 Class Attributes

**Table 5A-25.2 Position Controller Object Class Attributes**

| Number | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Get | Revision | UINT | Revision of this object. | The current value assigned to this attribute is two (02). |
| 2 thru 7 | These class attributes are either optional or conditional and are described in Chapter 4 of this specification. | | | | | |

## 5A-25.3 Instance Attributes

**Table 5A-25.3 Position Controller Object Instance Attributes**

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Get | Number of Attributes | USINT | Returns the total number of attributes supported by this object in this device. | Return value is in the range of 0 to 255. |
| 2 | Required | Get | Attribute List | Array of USINT | Returns an array with a list of the attributes supported by this object in this device. | Array size is defined by attribute 1. |
| 3 | Optional | Set | Mode | USINT | Operating mode. | 0 = Position mode(default),<br>1 = Velocity mode,<br>2 = Torque mode. |
| 4 | Optional | Set | Position Units | DINT | Position Units ratio value is the number of actual position feedback counts equal to one position unit. | Set this value to a positive number only. Default = 1. |
| 5 | Optional | Set | Profile Units | DINT | Profile Units ratio value is the number of actual position feedback counts per second or second$^2$ equal to one velocity, acceleration or deceleration unit. | This value is set to a positive number only. Default = 1. |
| 6 | Required | Set | Target Position | DINT | Profile move position defined in position Units | This value can be in the range of 0x80000001 to 0x7FFFFFFF. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Position Controller Object, Class Code: $25_{Hex}$

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 7 | Required | Set | Target Velocity | DINT | Profile velocity defined in profile units per second. | This value is set to a positive number only. |
| 8 | Required | Set | Acceleration | DINT | Profile Acceleration rate defined in profile units per second$^2$. | This value is set to a positive number only. |
| 9 | Optional | Set | Deceleration | DINT | Profile Deceleration rate defined in profile units per second$^2$. | This value is set to a positive number only. |
| 10 | Optional | Set | Incremental Position Flag | BOOL | Incremental Position Flag | If set to 0 the target position (attribute 6) will be interpreted as absolute. If set to 1 the target position will be interpreted as incremental |
| 11 | Required | Set | Load Data/ Profile Handshake | BOOL | Used to Load Command Data, Start a Profile Move, and indicate that a Profile Move is in progress. | See “Semantics” at end of this table. |
| 12 | Optional | Get | On Target Position | BOOL | On target position flag indicates that the motor is within the deadband (Attribute 38) distance to the target position. | Reads Set when the Target position equals the actual position within deadband limits. Will clear if target position is changed. |
| 13 | Optional | Set | Actual Position | DINT | Actual Absolute position value equals the real position in position units. Set to re-define Actual Position | When set this value can be in the range of 0x80000001 to 0x7FFFFFFF. |
| 14 | Optional | Get | Actual Velocity | DINT | Actual Velocity in profile units/sec. | Value read will be positive. |
| 15 | Optional | Get | Commanded Position | DINT | This value equals the instantaneous calculated position. | When read this value will be in the range of 0x80000001 to 0x7FFFFFFF. |
| 16 | Optional | Get | Commanded Velocity | DINT | This value equals the Instantaneous calculated velocity in profile units per second. | Value read will be positive. |
| 17 | Optional | Set | Enable | BOOL | Enable Output. | Set to enable drive and feedback, clear to disable. |
| 18 | Optional | Set | Profile Type | USINT | Profile Type code defines the type of move profile. | 0 = Trapezoidal,<br>1 = S-Curve,<br>2 = Parabolic.<br>3 - 127 = Reserved by CIP<br>128 - 255 = Vendor specific |
| 19 | Optional | Set | Profile Gain | DINT | This attribute provides a gain value for non-trapezoidal profiles such as S-Curve profiling. The implementation and function of this gain is vendor specific. | Range is defined by the vendor. |
| 20 | Optional | Set | Smooth Stop | BOOL | Smooth stop motor. | Set to force immediate deceleration to zero velocity at programmed decel rate. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Position Controller Object, Class Code: $25_{Hex}$

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 21 | Optional | Set | Hard Stop | BOOL | Hard stop motor. | Set to force immediate deceleration to zero velocity at max decel rate. |
| 22 | Optional | Set | Jog Velocity | DINT | Defines the jogging velocity in profile units per second. | This value is set to a positive number only. |
| 23 | Optional | Set | Direction | BOOL | Instantaneous direction | 0 = negative or reverse direction and 1 = positive or forward. This value can be set in Velocity mode to change direction. |
| 24 | Optional | Set | Reference Direction | BOOL | Defines direction. | 0 = forward direction is clockwise,<br>1 = reverse direction is counter clockwise as viewed from the load end of the motor shaft. |
| 25 | Optional | Set | Torque | DINT | Output torque. | Set this value to change the torque (Torque mode only) or read the current torque. 0 = no torque output. Range defined by the vendor. |
| 26 | Optional | Set | Positive Torque Limit | DINT | This value sets the maximum allowable torque output in the positive direction. | This value is set to a positive number only. Range defined by the vendor. |
| 27 | Optional | Set | Negative Torque Limit | DINT | This value sets the maximum allowable torque output in the negative direction. | This value is set to a negative number only. Range defined by the vendor. |
| 28 | | | Reserved | | | |
| 29 | Optional | Get | Wrap Around | BOOL | Position Wrap Around indicator Flag | If set to 1 the motor has gone past its maximum position. This can only happen in Velocity mode and is not necessarily a fault condition. This bit is reset when read. |
| 30 | Optional | Set | Kp | INT | Proportional gain. | Range is 0 to 32767. |
| 31 | Optional | Set | Ki | INT | Integral gain. | Range is 0 to 32767. |
| 32 | Optional | Set | Kd | INT | Derivative gain. | Range is 0 to 32767. |
| 33 | Optional | Set | MaxKi | INT | Integration limit. | Range is 0 to 32767. |
| 34 | Optional | Set | KiMode | USINT | Integration limit Mode. | 0 = use Ki term at all times,<br>1 = use Ki term only when stopped and holding position. |
| 35 | Optional | Set | Velocity Feed Forward. | INT | Velocity feed forward gain value. | Range is 0 to 32767. |
| 36 | Optional | Set | Accel Feed Forward | INT | Acceleration feed forward gain value. | Range is 0 to 32767. |
| 37 | Optional | Get | Sample Rate | INT | Update sample rate in µSeconds. | Value returned is positive. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Position Controller Object, Class Code: $25_{Hex}$

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 38 | Optional | Set | Position Deadband | USINT | Set this value to prevent axis hunting within the desired window. | Range is 0 to 255. |
| 39 | Optional | Set | Feedback Enable | BOOL | This flag will set or clear automatically with the Enable attribute (17). Feedback can be turned off, using this attribute, leaving the enable on, for offset adjustments on the drive unit | 0 = command output generator off,<br>1 = command output generator on. |
| 40 | Optional | Set | Feedback Resolution | DINT | Feedback resolution in counts. Number of actual position feedback counts in one revolution of the position feedback device. | This value is set to a positive number only. |
| 41 | Optional | Set | Motor Resolution | DINT | Motor resolution in motor steps. Number of motor steps in one revolution of the motor. | This value is set to a positive number only. |
| 42 | Optional | Set | Position Tracking Gain | DINT | Position Tracking Gain (Stepper) Gain value for position maintenance to control steppers with position feedback. | Range defined by the vendor. |
| 43 | Optional | Set | Max Correction velocity | UINT | Position maintenance value to prevent stepper motor stalls. Value in counts per second. | This value is set to a positive number only. |
| 44 | Optional | Set | Max Static Following Error | DINT | Maximum allowable following error when the motor is stopped and holding position. If the difference between actual and commanded position exceeds this value, following error flag is set. | Set to value to a positive number only. |
| 45 | Optional | Set | Max Dynamic Following Error | DINT | Maximum allowable following error when the motor is in motion. If difference between actual and commanded position exceeds this value, following error flag is set. | Set to value to a positive number only. |
| 46 | Optional | Set | Following Error Action | USINT | Following error action code. | 0 = Command Output Generator Off,<br>1 = Hard Stop,<br>2 = Smooth stop<br>3 = no action.<br>4 - 127 = Reserved by CIP<br>128 - 255 = Vendor specific |
| 47 | Optional | Set | Following Error Fault | BOOL | Following error occurrence flag. Set when a following error occurs. This bit is reset when another move is attempted. | Set when a following error occurs. This bit can be cleared directly or by re-programming the Following Error Action attribute. |
| 48 | Optional | Get | Actual Following Error | DINT | Actual Following Error. | This value is the actual amount of Following Error in position feedback counts. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 49 | Optional | Set | Hard Limit Action | USINT | Hard limit action code. | 0 = Command Output Generator Off<br>1 = Hard Stop<br>2 = smooth stop.<br>3 - 127 = Reserved by CIP<br>128 - 255 = Vendor specific |
| 50 | Optional | Get | Forward Limit | BOOL | Motion is not allowed in the positive direction when active. | Set when the forward limit stop is active. |
| 51 | Optional | Get | Reverse Limit | BOOL | Motion is not allowed in the negative direction when active. | Set when a reverse limit stop is active. |
| 52 | Optional | Set | Soft Limit Enable | BOOL | Enables soft limits | When set, motion that exceeds the defined limits will result in a motor stop. |
| 53 | Optional | Set | Soft Limit Action | USINT | Soft limit action code. | 0 = Command Output Generator Off,<br>1 = Hard Stop,<br>2 = smooth stop. Action codes 128 through 255 are for vendor specific action. |
| 54 | Optional | Set | Positive Soft Limit Position | DINT | Soft limit positive boundary defined in position units.. | This value can be in the range of 0x80000001 to 0x7FFFFFFF. |
| 55 | Optional | Set | Negative Soft Limit Position | DINT | Soft limit negative boundary defined in position units.. | This value can be in the range of 0x80000001 to 0x7FFFFFFF. |
| 56 | Optional | Get | Positive Limit Triggered | BOOL | Hard Forward limit occurrence flag. | Set when a positive limit stop occurs. |
| 57 | Optional | Get | Negative Limit Triggered | BOOL | Hard Reverse limit occurrence flag. | Set when a negative limit stop occurs. |
| 58 | Required | Get | Load Data Complete | BOOL | Indicates that valid data for a valid I/O command message type has been loaded into the position controller device. | See “Semantics” at the end of this table. |

### 5A-25.3.1 Semantics

#### 5A-25.3.1.1 Profile in Progress

This attribute performs three functions:

- loading data in the I/O command message;
- starting a Profile Move in the both the i/o command message and explicit messaging; and
- indicating if a Profile Move is in process. See the chart below for a complete explanation of the device's behavior with respect to this bit.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-25.4 Behavior of Device With Respect to Profile In Progress Attribute**

| Connection | Message Type or Service | Bit Name | Behavior |
|---|---|---|---|
| I/O | Command | Load Data/ Start Profile | When this bit transitions from zero to one, the position controller device will attempt to load the data contained in the command message data bytes. If the command message type contained in the command message is the command message type that starts a Profile Move, the Profile Move will start. See the table in Section 6-18.4.3 for an explanation of which command message type starts a Profile Move for each mode. |
| | Response | Profile in Progress | This bit will indicate that a Profile Move is in progress. This bit will read 1 when a Profile Move is started and will remain 1 until the Profile Move is complete or terminated, at which point it will be zero. |
| Explicit | Set_Attribute _Single | Start Profile | A "Set_Attribute_Single" service, which sets this attribute to 1 will start a Profile Move. A "Set_Attribute_Single" service, which sets this attribute to 0 will have no effect. |
| | Get_Attribute _Single | Profile in Progress | This bit will indicate that a Profile Move is in progress. This bit will be one after receipt of a set attribute service which sets this attribute to 1 and will remain 1 until the profile move is complete or terminated, at which point it will be zero. |

#### 5A-25.3.1.2 Load Data Complete

This bit is used for data handshaking in the I/O message. It reflects that the position controller device has successfully loaded the data contained in the I/O command message data bytes. This attribute will be reset when the Load Command Data/Start Profile bit is reset. Refer to the Position Controller Device Profile in Chapter 6 for an explanation of the handshaking procedure. This bit is not affected by explicit messaging.

### 5A-25.4 Common Services

**Table 5A-25.5 Position Controller Object Common Services**

| Need in Implementation | Name | Service Code | Description of Service |
|---|---|---|---|
| Required | Get_Attribute_Single | 0Ehex | Returns the contents of the specified attribute |
| Required | Set_Attribute_Single | 10hex | Modifies the attribute value. |
| See Appendix A for definition of these services | | | |

### 5A-25.5 Object-specific Services

The Position Controller object provides no object-specific services.

### 5A-25.6 Behavior

Object behavior is defined by the following state diagrams

#### 5A-25.6.1 Position Controller State Diagrams

#### 5A-25.6.1.1 Profile Move Generation State Diagrams

The state diagram below describes Position Controller profile generation. The Profile generator uses Acceleration, Target, Velocity and Deceleration to perform a profile move to the Target Position. In profile velocity mode the Target Position is infinite with polarity defined by the direction attribute until such time the device is commanded to decelerate and stop.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

After the profile position is generated, it passes through a limit filter. If the generated Profile position is outside the defined software limits, or if a hardware limit is active, the profile is modified and the output Command position is limited. The limit function and attributes are optional.

The Command Position is then sent to the output generator, which produces a control signal. The output generator can be servo, stepper or some other method for controlling position. Commanded Acceleration and Velocity are also sent to the output generator for feed forward purposes. Feedback is optional unless required by the output method being used.

**Figure 5A-25.1 Position Controller Profile Generation State Diagram**

### 5A-25.6.1.2 Servo Output Generation State Diagram

The diagram below is an example of a Servo Command output generator. Commanded Position, Velocity and Acceleration input comes from the Profile Generation diagram.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5A-25.2 Servo Output Generator State Diagram**

### 5A-25.6.1.3 Torque Mode Output State Diagram

The following diagram describes the direct torque mode function.

**Figure 5A-25.3 Torque Mode State Diagram**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-26 Block Sequencer Object

**Class Code: 26 hex**

This object handles the execution of Command Blocks or Command Block chains.

## 5A-26.1 Class Attributes

**Table 5A-26.1 Block Sequencer Object Class Attributes**

| Number | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |

## 5A-26.2 Instance Attributes

These attributes can be used for control, configuration or status.

**Table 5A-26.2 Block Sequencer Object Instance Attributes**

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Set/Get | Block | USINT | Instance number of starting Command Block. | This value Defines the Command Block instance to execute. Set from 1 to 255. |
| 2 | Required | Get/Set | Block Execute | BOOL | Block execution flag. | Setting this value executes the block defined by the Block Attribute (1). When this value reads back cleared, the block or chain of blocks is done. |
| 3 | Required | Get | Current Block | USINT | Current block in execution. | This value contains the Command Block instance number of the currently executing block (1 - 255). |
| 4 | Required | Get | Block Fault | BOOL | Block fault flag. | Set when a block error occurs, such as the Wait Equals command time-out or execution of an invalid Command Block. when a Block Fault occurs block execution will stop. This bit is reset when the Block Fault Code attribute (5) is read. |
| 5 | Optional | Get | Block Fault Code | USINT | Block fault Code. | Defines the specific block fault.<br>0 = no fault,<br>1 = invalid or empty block data,<br>2 = command time-out (Wait Equals) ,<br>3 = execution fault. |
| 6 | Optional | Set | Counter | DINT | Sequencing Counter. | Must be positive. Counter that can be used for sequencing loops. |

## 5A-26.3 Common Services

**Table 5A-26.3 Block Sequencer Object Common Services**

| Need in Implementation | Name | Service Code | Description of Service |
|---|---|---|---|
| Required | Get_Attribute_Single | 0Ehex | Returns the contents of the specified attribute |
| Required | Set_Attribute_Single | 10hex | Modifies the attribute value. |
| See Appendix A for a description of these services | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-26.4 Object-specific Services

The Block Sequencer object provides no object-specific services.

## 5A-26.5 Behavior

Object behavior is defined by the following state diagrams

### 5A-26.5.1 Block Sequencer State Diagrams

The diagram below describes the Block Sequencer. When the Sequencer is commanded to execute a block, the Block Execution state is entered. Block execution continues on consecutive blocks until the end of the linked chain is reached or an error occurs.

**Figure 5A-26.1 Block Sequencer State Diagram**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-27 Command Block Object

**Class Code: 27 hex**

Each instance of the Command Block object defines a specific command. These blocks can be linked to other blocks to form a command block chain.

## 5A-27.1 Class Attributes

**Table 5A-27.1 Command Block Object Class Attributes**

| Number | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |

## 5A-27.2 Attributes

These attributes are downloaded at configuration time and executed at run time through the Block Sequencer object. Attributes 3, 4, and 5 definitions are dependent on the block command attribute (01). Each instance of the Block command class defines a different block command that can be linked to any other block command instance to form an execution sequence.

**Table 5A-27.2 Command Block Object Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Set | Block Command | USINT | Block Command # | Defines the format of the block data. Command data formats are defined below. |
| 2 | Required | Set | Block Link # | USINT | Block link instance number. | This value provides a link to the next block instance to execute. When this block is done, the link block will be executed. |
| 3 | Depends on Command | Set | Depends on Command | Depends on Command | | Refer to the command definitions for the description of attribute 3. |
| 4 | Depends on Command | Set | Depends on Command | Depends on Command | | Refer to the command definitions for the description of attribute 4. |
| 5 | Depends on Command | Set | Depends on Command | Depends on Command | | Refer to the command definitions for the description of attribute 5. |
| 6 | Depends on Command | Set | Depends on Command | Depends on Command | | Refer to the command definitions for the description of attribute 6. |
| 7 | Depends on Command | Set | Depends on Command | Depends on Command | | Refer to the command definitions for the description of attribute 6. |

## 5A-27.3 Command Specific Attribute Services

This section defines the Command block data. Command blocks can be linked together to form a command block chain. Looping and branching commands are supported.

### 5A-27.3.1 Modify Attribute Command - 01

This command is used to change an attribute's value. Attribute 1 must be set to 01.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-27.3 Command Block - Modify Attribute Command**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 3 | Required for Command 01 | Set | Target Class | USINT | Target class number to perform block sequencing on. | This value Defines the class which will be sequenced. |
| 4 | Required for Command 01 | Set | Target Instance | USINT | Target instance number of the class to perform block sequencing on. | This value Defines the instance of the class which will be sequenced. |
| 5 | Required for Command 01 | Set | Attribute # | USINT | Position Controller Attribute | Position Controller class attribute number. Must be a settable attribute. |
| 6 | Required for Command 01 | Set | Attribute Data | Dependent on attribute # | Attribute Data. | The new Attribute data. |

### 5A-27.3.2 Wait Equals Command - 02

This command is used to stop execution of a linked chain of command until an attribute becomes valid. Attribute 1 must be set to 02.

**Table 5A-27.4 Command Block – Wait Equals Command**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 3 | Required for Command 02 | Set | Target Class | USINT | Target class number to perform block sequencing on. | This value Defines the class which will be sequenced. |
| 4 | Required for Command 02 | Set | Target Instance | USINT | Target instance number of the class to perform block sequencing on. | This value Defines the instance of the class which will be sequenced. |
| 5 | Required for Command 02 | Set | Attribute # | USINT | Position Controller Attribute | Position Controller class attribute number. Must be a settable attribute. |
| 6 | Required for Command 02 | Set | Compare Time-Out value | DINT | Compare Time-out value in milliseconds. | Set from 0 to 7FFFFFFF hex. If compare does not happen within time-out a fault is generated and motion stops. 0 = no time-out. |
| 7 | Required for Command 02 | Set | Compare Data | Dependent on attribute # | Compare data for end of command. | If the attribute listed above is Becomes equal to the compare data the block is done and the next link block is executed. |

### 5A-27.3.3 Conditional Link Greater Than Command - 03

This command is used for conditional linking or branching in a linked chain of commands. Attribute 1 must be set to 03.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-27.5 Command Block – Conditional Link Greater Than Command**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 3 | Required for Command 03 | Set | Target Class | USINT | Target class number to perform block sequencing on. | This value Defines the class which will be sequenced. |
| 4 | Required for Command 03 | Set | Target Instance | USINT | Target instance number of the class to perform block sequencing on. | This value Defines the instance of the class which will be sequenced. |
| 5 | Required for Command 03 | Set | Attribute # | USINT | Position Controller Attribute | Position Controller class attribute number. Must be a settable attribute. |
| 6 | Required for Command 03 | Set | Compare Link # | USINT | Conditional Link Number. | Alternate Link block if attribute is greater than compare data. |
| 7 | Required for Command 03 | Set | Compare Data | Dependent on attribute # | Compare data for conditional link. | If the attribute listed above is greater than the compare data the normal link attribute (02) is ignored and the next block executed is the compare link block. |

### 5A-27.3.4 Conditional Link Less Than Command - 04

This command is used for conditional linking or branching in a linked chain of commands. Attribute 1 must be set to 04.

**Table 5A-27.6 Command Block – Conditional Link Less Than Command**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 3 | Required for Command 04 | Set | Target Class | USINT | Target class number to perform block sequencing on. | This value Defines the class which will be sequenced. |
| 4 | Required for Command 04 | Set | Target Instance | USINT | Target instance number of the class to perform block sequencing on. | This value Defines the instance of the class which will be sequenced. |
| 5 | Required for Command 04 | Set | Attribute # | USINT | Position Controller Attribute | Position Controller class attribute number. Must be a settable attribute. |
| 6 | Required for Command 04 | Set | Compare Link # | USINT | Conditional Link Number. | Alternate Link block if attribute is less than compare data. |
| 7 | Required for Command 04 | Set | Compare Data | Dependent on attribute # | Compare data for conditional link. | If the attribute listed above is less than the compare data the normal link attribute (02) is ignored and the next block executed is the compare link block. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-27.3.5 Decrement Counter Command - 05

This command is used to decrement the Block Sequencers counter attribute used for looping. Attribute 1 must be set to 05.

**Table 5A-27.7 Command Block – Decrement Counter Command**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| No additional attributes required for this command | | | | | | |

### 5A-27.3.6 Delay Command - 06

This command is used to perform a delay in a linked chain of commands. Attribute 1 must be set to 06.

**Table 5A-27.8 Command Block – Delay Command**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 3 | Required for Command 06 | Set | Delay | DINT | Delay in Milliseconds | Set the delay in milliseconds 1 hex to 7FFFFFFF hex |

### 5A-27.3.7 Trajectory Command - 07

This command is used to initiate a move. Attribute 1 must be set to 07.

**Table 5A-27.9 Command Block – Trajectory Command**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 3 | Required for Command 07 | Set | Target Position | DINT | Target Position. | Profile destination defined in position Units |
| 4 | Required for Command 07 | Set | Target Velocity | DINT | Target Velocity. | Profile velocity defined in profile units per second. |
| 5 | Required for Command 07 | Set | Incremental | BOOL | Absolute / Incremental flag. | 0 = absolute position,<br>1 = incremental position |

### 5A-27.3.8 Trajectory Command and Wait - 08

This command is used to initiate a move and wait for completion. Attribute 1 must be set to 08.

**Table 5A-27.10 Command Block – Trajectory Command and Wait**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 3 | Required for Command 08 | Set | Target Position | DINT | Target Position. | Profile destination defined in position Units |
| 4 | Required for Command 08 | Set | Target Velocity | DINT | Target Velocity. | Profile velocity defined in profile units per second. |
| 5 | Required for Command 08 | Set | Incremental | BOOL | Absolute / Incremental flag. | 0 = absolute position,<br>1 = relative position |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-27.3.9 Velocity Change Command - 09

This command is used to initiate a move and wait for completion. Attribute 1 must be set to 09.

**Table 5A-27.11 Command Block – Change Command**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 3 | Required for Command 09 | Set | Target Velocity | DINT | Target Velocity. | Profile velocity defined in profile units per second. |

### 5A-27.3.10 Goto Home Command - 10

This command is used perform a move to the captured Home position. Attribute 1 must be set to 10.

**Table 5A-27.12 Command Block – Goto Home Command**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 3 | Required for Command 10 | Set | Home Offset | DINT | Home Position Offset | The offset plus the captured Home position equals the absolute target position. |
| 4 | Required for Command 10 | Set | Velocity | DINT | Target Velocity. | Profile velocity defined in profile units per second. |

### 5A-27.3.11 Goto Index Command - 11

This command is used perform a move to the captured Index position. Attribute 1 must be set to 11.

**Table 5A-27.13 Command Block – Goto Index Command**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 3 | Required for Command 11 | Set | Index Offset | DINT | Index Position Offset | The offset plus the captured Index position equals the absolute target position. |
| 4 | Required for Command 11 | Set | Velocity | DINT | Target Velocity. | Profile velocity defined in profile units per second. |

### 5A-27.3.12 Goto Registration Position Command - 12

This command is used perform a move to the captured Registration position. Attribute 1 must be set to 12.

**Table 5A-27.14 Command Block – Goto Registration Position Command**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 3 | Required for Command 12 | Set | Registration Offset | DINT | Registration Position Offset | The offset plus the captured Registration position equals the absolute target position. |
| 4 | Required for Command 12 | Set | Velocity | DINT | Target Velocity. | Profile velocity defined in profile units per second. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-27.3.13 Common Services

**Table 5A-27.15 Command Block Common Services**

| Need in Implementation | Name | Service Code | Description of Service |
|---|---|---|---|
| Required | Get_Attribute_Single | 0E hex | Returns the contents of the specified attribute |
| Required | Set_Attribute_Single | 10 hex | Modifies the attribute value. |
| See Appendix A for definitions of these services | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-28 Motor Data Object

**Class Code: 28 hex**

This object serves as a database for motor parameters.

## 5A-28.1 Class Attributes

**Table 5A-28.1 Motor Data Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |

## 5A-28.2 Instance Attributes

The Motor Data Instance Attribute set varies depending on the type of motor that the object is representing. Instance attribute 3 “MotorType” is required, and its value determines the motor specific attributes that are available for that motor type. For all motor types, Attributes 1-5 are the same.

**Table 5A-28.2 Motor Data Instance Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|---|
| 1 | Conditional | Get | NV | NumAttr | USINT | Number of Attributes supported |
| 2 | Optional | Get | NV | Attributes | Array of USINT | List of attributes supported |
| 3 | Required | Set/Get | NV | MotorType | USINT | 0 - Non-standard motor<br>1 - PM DC Motor<br>2 - FC DC Motor<br>3 - PM Synchronous Motor<br>4 - FC Synchronous Motor<br>5 - Switched Reluctance Motor<br>6 - Wound Rotor Induction Motor<br>7 - Squirrel Cage Induction Motor<br>8 - Stepper Motor<br>9 - Sinusoidal PM BL Motor<br>10 - Trapezoidal PM BL Motor |
| 4 | Optional | Set/Get | NV | CatNumber | SHORT_ STRING | Manufacturer's Motor Catalog Number (Nameplate number) 32 chars max |
| 5 | Optional | Set/Get | NV | Manufacturer | SHORT_ STRING | Manufacturer's Name 32 chars max |

### 5A-28.2.1 Motor Type Specific Motor Data Instance Attributes

Different motor types require different data to describe the motor. For example, AC Induction motors do not need field current data like a DC motor to describe the motor. For this reason, motor data attributes that are numbered greater than 5 are described separately for different classes of motors. The following table shows the classes of motors described in this specification. Other motor classes and types will be described in future revisions of this specification.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-28.3 Motor Type Specific Motor Data Instance Attributes**

| Motor Class | Motor Types in Class | Section Reference |
|---|---|---|
| AC Motor | 3 - PM Synchronous<br>6 - Wound Rotor Induction<br>7 - Squirrel Cage Induction Motor | 5A-28.2.1.1 |
| DC Motor | 1 - PM DC Motor<br>2 - FC DC Motor | 5A-28.2.1.2 |

### 5A-28.2.1.1 AC Motor Instance Attributes

**Table 5A-28.4 AC Motor Instance Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|---|
| 6 | Required | Set/Get | NV | RatedCurrent | UINT | Rated Stator Current Units: [100mA] |
| 7 | Required | Set/Get | NV | RatedVoltage | UINT | Rated Base Voltage Units: [V] |
| 8 | Optional | Set/Get | NV | RatedPower | UDINT | Rated Power at Rated Freq Units: [W] |
| 9 | Optional | Set/Get | NV | RatedFreq | UINT | Rated Electrical Frequency Units: [Hz] |
| 10 | Optional | Set/Get | NV | RatedTemp | UINT | Rated Winding Temperature<br>Units: [ degrees C] |
| 11 | Optional | Set/Get | NV | MaxSpeed | UINT | Maximum allowed motor speed Units: [RPM] |
| 12 | Optional | Set/Get | NV | PoleCount | UINT | Number of poles in the motor. |
| 13 | Optional | Set/Get | NV | TorqConstant | UDINT | Motor torque constant Units: [0.001 x Nm/A] |
| 14 | Optional | Set/Get | NV | Inertia | UDINT | Rotor Inertia Units: [$10^{-6}$ x kg.$m^2$] |
| 15 | Optional | Set/Get | NV | BaseSpeed | UINT | Nominal speed at rated frequency from nameplateUnits: [RPM] |
| 19 | Optional | Set/Get | NV | ServiceFactor | USINT | Units: [%] Range: 0 .. 255 |
| 20 | Optional | Get | NV | International CatNumber | STRINGI | Catalog Name or Number |
| 21 | Optional | Get | NV | Internationl Manufacturer Name | STRINGI | Name of Motor Manufacturer |
| 22 | Optional | Set | NV | SerialNumber | SHORT STRING | Name Plate Serial Number of Motor (32 characters or less) |
| 23 | Optional | Set | NV | TagName | SHORT STRING | Plant's Tag Name of Motor (32 characters or less) |
| 24 | Optional | Set | NV | SupplyStyle | BOOL | 0 = three phase, 1 = single phase |
| 25 | Optional | Set | NV | TimeRating | UINT | Maximum Continuous Operation Time (in minutes) |
| 26 | Optional | Set | NV | InrushCurrent | UINT | Maximum inrush current, Units: [100mA] |
| 27 | Optional | Set | NV | LockedRotor CodeLetter | SHORT STRING | NEMA MG-1 Code Letter [MG1-10.37.2]<br>The single ASCII character corresponding to the Code Letter for Locked Rotor kVA |
| 28 | Optional | Set | NV | DesignLetter | SHORT STRING | IEC or NEMA Design Letters |
| 29 | Optional | Set | NV | Thermal Protection | BOOL | TRUE when motor is marked "Thermally Protected" |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-28.2.1.2 DC Motor Instance Attributes

**Table 5A-28.5 DC Motor Instance Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|---|
| 6 | Required | Set/Get | NV | RatedCurrent | UINT | Rated Armature Current<br>Units: [100mA] |
| 7 | Required | Set/Get | NV | RatedVoltage | UINT | Rated Armature Voltage<br>Units: [V] |
| 8 | Optional | Set/Get | NV | RatedPower | UDINT | Rated Power at MaxSpeed<br>Units: [W] |
| | | | | | | |
| 10 | Optional | Set/Get | NV | RatedTemp | UINT | Rated Winding Temperature<br>Units: [ degrees C] |
| 11 | Optional | Set/Get | NV | MaxSpeed | UINT | Maximum allowed motor speed<br>Units: [RPM] |
| 13 | Optional | Set/Get | NV | TorqConstant | UDINT | Motor torque constant<br>Units: [0.001 x Nm/A] |
| 14 | Optional | Set/Get | NV | Inertia | UDINT | Rotor Inertia<br>Units: [$10^{-6}$ x kg.m$^2$] |
| 15 | Optional | Set/Get | NV | BaseSpeed | UINT | Nominal speed at rated voltage<br>Units: [RPM] |
| 16 | Optional | Set/Get | NV | RatedFieldCur | UDINT | Rated Field Current<br>Units: [mA] |
| 17 | Optional | Set/Get | NV | MinFieldCur | UDINT | Minimum Field Current<br>Units: [mA] |
| 18 | Optional | Set/Get | NV | RatedFieldVolt | UINT | Rated Field Voltage<br>Units: [V] |
| 20 | Optional | Get | NV | International CatNumber | STRINGI | Catalog Name or Number |
| 21 | Optional | Get | NV | Internationl Manufacturer Name | STRINGI | Name of Motor Manufacturer |
| 22 | Optional | Set | NV | SerialNumber | SHORT STRING | Name Plate Serial Number of Motor (32 characters or less) |
| 23 | Optional | Set | NV | TagName | SHORT STRING | Plant's Tag Name of Motor (32 characters or less) |
| 24 | Optional | Set | NV | SupplyStyle | BOOL | 0 = three phase, 1 = single phase |
| 25 | Optional | Set | NV | TimeRating | UINT | Maximum Continuous Operation Time (in minutes) |
| 26 | Optional | Set | NV | InrushCurrent | UINT | Maximum inrush current, Units: [100mA] |
| 27 | Optional | Set | NV | LockedRotor CodeLetter | SHORT STRING | NEMA MG-1 Code Letter [MG1-10.37.2]<br>The single ASCII character corresponding to the Code Letter for Locked Rotor kVA |
| 28 | Optional | Set | NV | DesignLetter | SHORT STRING | IEC or NEMA Design Letters |
| 29 | Optional | Set | NV | Thermal Protection | BOOL | TRUE when motor is marked "Thermally Protected" |

The following engineering abbreviations are used in the above Motor Data Instance Attribute descriptions.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-28.6 Abbreviations Used in This Object**

| Abbreviation | Description |
|---|---|
| mA | milli Amps |
| V | Volts |
| RPM | Revolutions Per Minute |
| $Kg.m^2$ | kilograms times meters squared |
| Nm/A | Newton meters per Amp |
| Degrees C | degrees Centigrade |
| Hz | Hertz |
| W | Watts |

## 5A-28.3 Common Services

**Table 5A-28.7 Motor Data Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0Ehex | Conditional [1] | Required | Get_Attributes_Single | Returns the contents of the specified attribute. |
| 10hex | n/a | Required | Set_Attributes_Single | Modifies an attribute value. |
| 15hex | n/a | Optional | Restore | Restores attribute values from a storage location accessible by the Save service. Service is only available if the Control Supervisor Object of the drive is in the **Ready** or **Wait_Power** states. |
| 16hex | n/a | Optional | Save | Saves all attribute values to a location accessible by the Restore service. |

Table Footnotes:

1 The Get_Attribute_Single service is ***required*** at the class level if any class attributes are implemented

See Appendix A for definitions of these common services.

## 5A-28.4 Object–specific Services

The Motor Data object provides no object specific services.

## 5A-28.5 Behavior

The Motor Data Object serves only as an internal database for motor parameters, which are accessed by other objects in the drive.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-29 Control Supervisor Object

## Class Code: 29 Hex

This object models all the management functions for devices within the "Hierarchy of Motor Control Devices". The behavior of motor control devices is described by the State Transition Diagram and the State Event Matrix (see Section 5A-29.5).

## 5A-29.1 Class Attributes

**Table 5A-29.1Control Supervisor Object Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |

## 5A-29.2 Instance Attributes

**Table 5A-29.2Control Supervisor Object Instance Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|---|
| 1 | Optional | Get | | NumAtttr | USINT | Number of Attributes supported |
| 2 | Optional | Get | | Attributes | Array of USINT | List of attributes supported |
| 3 | Required | Set | | Run1 | BOOL | See Run/Stop Event Matrix |
| 4 | Optional | Set | | Run2 | BOOL | See Run/Stop Event Matrix |
| 5 | Optional | Set | | NetCtrl | BOOL | Requests Run/Stop control to be local or from network.<br>0 = Local Control<br>1 = Network Control<br>Note that the actual status of Run/Stop control is reflected in attribute 15, CtrlFromNet. |
| 6 | Optional | Get | | State | USINT | 0 = Vendor Specific<br>1 = Startup<br>2 = Not_Ready<br>3 = Ready<br>4 = Enabled<br>5 = Stopping<br>6 = Fault_Stop<br>7 = Faulted |
| 7 | Required for Drives and Servos only | Get | | Running1 | BOOL | 1 = (Enabled **and** Run1) **or**<br>(Stopping **and** Running1) **or**<br>(Fault_Stop **and** Running1)<br>0 = Other state |
| 8 | Optional | Get | | Running2 | BOOL | 1 = (Enabled **and** Run2) **or**<br>(Stopping **and** Running2) **or**<br>(Fault_Stop **and** Running2)<br>0 = Other state |
| 9 | Optional | Get | | Ready | BOOL | 1 = Ready **or** Enabled **or** Stopping<br>0 = Other state |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Control Supervisor Object, Class Code: $29_{Hex}$

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|---|
| 10 | Required for Drives and Servos only | Get | | Faulted | BOOL | 1 = Fault Occurred (latched)<br>0 = No Faults present |
| 11 | Optional | Get | | Warning | BOOL | 1 = Warning (not latched)<br>0 = No Warnings present<br>If warnings are not supported, this attribute should always be 0 |
| 12 | Required for Drives and Servos only | Set | | FaultRst | BOOL | 0->1 = Fault Reset<br>0 = No action |
| 13 | Optional | Get | | FaultCode | UINT | If in **Faulted** state, FaultCode indicates the fault that caused the transition to **Faulted** state. If multiple faults occurred simultaneously, the vendor chooses which to report, and the rest are lost. If not in **Faulted** state, FaultCode indicates the fault that caused the last transition to the **Faulted** state.<br>Power up state of fault code is vendor specific.<br>Fault codes for drives are different than fault codes for starters.<br>See appropriate device profile for fault codes. |
| 14 | Optional | Get | | WarnCode | UINT | Code word indicating warning present. If multiple warnings are present, the lowest code value is displayed.<br>Note that fault codes for drives and servos are different than fault codes for starters.<br>See appropriate device profile for fault codes. |
| 15 | Optional | Get | | CtrlFromNet | BOOL | Status of Run/Stop control source.<br>0=Control is local<br>1=Control is from network |
| 16 | Optional | Set | | NetFaultMode* | USINT | Action on loss of CIP Network<br>0 = Fault + Stop<br>1 = Ignore (Warning Optional)<br>2= Vendor specific |
| 17 | Optional | Set | | ForceFault/ Trip | BOOL | 0 ->1 = Force |
| 18 | Optional | Get | | ForceStatus | BOOL | 0 = Not Forced<br>Nonzero = Forced |
| 19 | Optional | Set | | Delay * | ITIME | Time delay before any fault or idle mode action occurs after the fault of idle event is detected. |
| 20 | Optional | Set | | NetIdleMode * | USINT | Mode on reception of CIP communication IDLE event – See semantics below. |
| 21 | Optional | Set | | ProtectMode * | USINT | Mode on detection of a motor protection event – See semantics below. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|---|
| 22 | Optional | Set | | CycleCount * | UDINT | Number of operations (motor starts) on the equipment. This value shall be zero at the time of manufacture. This value may be settable to other values. This attribute is incremented each time a motor start activity occurs (eg: the starter causes a motor to begin turning, change direction or speed) |
| 23 | Conditional | Set | NV | Fault/Warning Code Style | USINT | Identifies the error and warning codes produced by this instance of Control Supervisor<br>Default = 0 |

* these attribute values shall be maintained through power cycles

### 5A-29.2.1 Operation Mode Values

The three modes of operation attributes above (number 16, 20 and 21) describe the mode or action that the control supervisor shall assume after specific events. These modes of operation are also coupled with the Delay attribute. The attribute semantics of the NetFaultMode, NetIdleMode and ProtectionMode are as enumerated below:

**Table 5A-29.3 Operation Mode Values**

| Mode | Action | Error/Warning |
|---|---|---|
| 0 | Stop | Fault Code |
| 1 | Ignore (Warning Optional) | Warning Code optional |
| 2 | Vendor specific action | Error and Warning optional |
| 3 | Run 1 (Invalid for ProtectMode Attribute) | Fault Code |
| 4 | Run 2 (Invalid for ProtectMode Attribute) | Fault Code |
| 5 thru 99 | Reserved by CIP | |
| 100 thru 199 | Vendor specific action | Error and Warning optional |
| 200 thru 255 | Reserved by CIP | |

### 5A-29.2.2 NetFaultMode Attribute

The NetFaultMode attribute establishes the mode of operation (see Table 5A-29.3 Operation Mode Values) on loss of network communication. The default value for this attribute is zero. If the is attribute is not implemented, the behavior of the control supervisor shall be the “Stop” mode.

If the Delay attribute is both implement and non-zero, the mode specified will be active only after a time specified by the delay has expired.

### 5A-29.2.3 NetIdleMode Attribute

The NetIdleMode attribute establishes the mode of operation (see Table 5A-29.3 Operation Mode Values) on reception of network Idle communication. The default value for this attribute is zero. If this attribute is not implemented, the behavior of the control supervisor shall be the “Stop” mode.

If the Delay attribute is both implement and non-zero, the mode specified will be active only after a time specified by the delay has expired.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-29.2.4 ProtectMode Attribute

The ProtectMode attribute establishes the mode of operation (see Table 5A-29.3 Operation Mode Values) on reception of a motor protection event. The default value for this attribute is zero. If this attribute is not implemented, the behavior of the control supervisor shall be the "Stop" mode.

If the Delay attribute is both implement and non-zero, the mode specified will be active only after a time specified by the delay has expired.

### 5A-29.2.5 Fault/Warning Code Style Attribute

The Fault/Warning Code Style attribute is used to indicate the identification of which types of Fault and/or Warning codes are produced by this instance of the Control Supervisor. This attribute is optional except in instances where the Fault/Warning Codes utilized is not defined by the device profile or is different than the enumerations expected due to the reported device profile.

This attribute has the values shown in Table 5A-29.4.

**Table 5A-29.4 Fault/Warning Code Styles**

| Value | Meaning | Reference |
|---|---|---|
| 0 | Default value. The Fault and Warning Codes presented are those implied by the device profile. | Table 5A-29.5 |
| 1 | The DRIVECOM (16-bit) codes are utilized | Table 5A-29.6 |
| 2 | The Abbreviated (8-bit) codes are utilized | Table 5A-29.11 |
| 2 thru 255 | Reserved by CIP | |

The implied Fault and Warning Code usage broken down by device profile are shown in Table 5A-29.5.

**Table 5A-29.5 Implied Fault/Warning Code Usage by Device Profile**

| Value | Meaning | Reference |
|---|---|---|
| 02 hex | AC Drive | DRIVECOM (16-bit) code set |
| not assigned | Circuit Breaker | Abbreviated (8-bit) code set |
| not assigned | Circuit Disconnect | Abbreviated (8-bit) code set |
| 15 hex | Contactor | Abbreviated (8-bit) code set |
| 13 hex | DC Drive | DRIVECOM (16-bit) code set |
| not assigned | Fuse | Abbreviated (8-bit) code set |
| 03 hex | Motor Overload | Abbreviated (8-bit) code set |
| 16 hex | Motor Starter | Abbreviated (8-bit) code set |
| not assigned | Power Monitor | Abbreviated (8-bit) code set |
| not assigned | Servo Drive | DRIVECOM (16-bit) code set |
| 17 hex | Soft Starter | Abbreviated (8-bit) code set |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-29.3 Common Services

**Table 5A-29.6 Control Supervisor Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0E hex | Conditional | Required | Get_Attributes_Single | Returns the contents of the specified attribute. |
| 10 hex | n/a | Required | Set_Attributes_Single | Modifies an attribute value. |
| 05 hex | n/a | Required | Reset | Resets the drive to the **start-up** state. |
| See Appendix A for description of these services | | | | |

## 5A-29.4 Object Specific Services

The Control Supervisor object provides no object specific services.

## 5A-29.5 Behavior

The State Transition Diagram provides a graphical description of the states and corresponding state transitions. The State Event Matrix lists all events and the corresponding action to be taken while in each state. A subset of states and events may be supported in an application but the behavior must be consistent.

**Figure 5A-29.1Control Supervisor Object State Transition Diagram**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-29.7Control Supervisor Object State Event Matrix**

| Event | State | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **Non_Exist** | **Startup** | **Not_Ready** | **Ready** | **Enabled** | **Stopping** | **Fault_Stop** | **Faulted** |
| **Switch Off** | N/A | Transition to Non_Exist | Transition to Non_Exist | Transition to Non_Exist | Transition to Non_Exist | Transition to Non_Exist | Transition to Non_Exist | Transition to Non_Exist |
| **Switch on** | Transition to Startup | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| **Initialization Complete** | N/A | Transition to Not_Ready | N/A | N/A | N/A | N/A | N/A | N/A |
| **Main Power On** | N/A | N/A | Transition to Ready | N/A | N/A | N/A | N/A | N/A |
| **Run *** | N/A | N/A | N/A | Transition to Enabled | N/A | Transition to Enabled | N/A | N/A |
| **Stop *** | N/A | N/A | N/A | N/A | Transition to Stopping | N/A | N/A | N/A |
| **Stop Complete** | N/A | N/A | N/A | N/A | N/A | Transition to Ready | N/A | N/A |
| **Reset** | N/A | N/A | Transition to Startup | Transition to Startup | Transition to Startup | Transition to Startup | Transition to Startup | Transition to Startup |
| **Main Power Off** | N/A | N/A | N/A | Transition to Not_Ready | Transition to Faulted | Transition to Faulted | Transition to Faulted | N/A |
| **Fault Detected** | N/A | Transition to Faulted | Transition to Faulted | Transition to Faulted | Transition to Fault_Stop | Transition to Fault_Stop | N/A | N/A |
| **Fault_Stop Complete** | N/A | N/A | N/A | N/A | N/A | N/A | Transition to Faulted | N/A |
| **Fault Reset** | N/A | N/A | N/A | N/A | N/A | N/A | N/A | Transition to Not_Ready |

* See section 5A-29.5.1 for further explanation of these events and how they are generated.

### 5A-29.5.1 Run/Stop Event Matrix

Attribute 5, NetCtrl is used to request that Run Stop events be controlled from the network. The device however, has the option of inhibiting Run Stop events from the network, as the user/application may not allow Run Stop control from the network under certain circumstances. Only when attribute 15, CtrlFromNet is set to 1 by the device in response to a NetCtrl request, is Run Stop control actually accomplished from the network.

If attribute 15, CtrlFromNet is 1, the events Run and Stop are triggered by a combination of the Run1 and Run2 attributes as shown in the following table. Note that Run1 and Run2 have different contexts for different device types. The following table shows the Run1 and Run2 contexts for the devices within the motor control hierarchy.

**Table 5A-29.8 Run1/Run2 Contexts**

| | Starters | | | | | Drives and Servos |
|---|---|---|---|---|---|---|
| | **Contactor** | **Starter** | **Reverser** | **2 Speed** | **SoftStart** | |
| Run1 | Close | Run | RunFwd | RunLo | RunRamp1 | RunFwd |
| Run2 | NA | NA | RunRev | RunHigh | RunRamp2 | RunRev |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

If CtrlFromNet is 0, Run and Stop events must be controlled using local input(s) provided by the vendor. The Run and Stop events effect drive behavior as shown in the State Transition Diagram and the State Event Matrix.

**Table 5A-29.9 Run1 and Run2 Triggers**

| Run1A | Run2 | Trigger Event | Run Type |
|---|---|---|---|
| 0 | 0 | Stop | NA |
| 0 -> 1 | 0 | Run | Run1 |
| 0 | 0 -> 1 | Run | Run2 |
| 0 -> 1 | 0 -> 1 | No Action | NA |
| 1 | 1 | No Action | NA |
| 1->0 | 1 | Run | Run2 |
| 1 | 1->0 | Run | Run1 |

**Important note:** Local stop and run signals could override or be interlocked with the run/stop control through CIP. These are vendor specific features. Vendors should explain these interlocks and overrides in their product documentation.

## 5A-29.6 Fault and Warning codes

This table lists the fault and warning codes used by AC Drives, DC Drives, and Servos for the 'Control Supervisor' object. The source of the fault codes is the DRIVECOM Nutzergruppe e.V, which has granted permission to ODVA to use the fault codes in ODVA device profiles.

**Table 5A-29.10 DRIVECOM (16-bit) Fault and Warning Codes**

| Code Value [Hex] | Meaning |
|---|---|
| **0000** | **No fault** |
| | |
| **1000** | **General Fault** |
| | |
| **2000** | **Current** |
| 2100 | Current, Device Input Side |
| 2110 | Short Circuit/Short to Earth |
| 2120 | Short to Earth |
| 2121 | Short to earth in Phase L1 |
| 2122 | Short to earth in Phase L2 |
| 2123 | Short to earth in Phase L3 |
| 2130 | Short Circuit |
| 2131 | Short Circuit in Phases L1 -L2 |
| 2132 | Short Circuit in Phases L2-L3 |
| 2133 | Short Circuit in Phases L3-L1 |
| 2200 | Current Inside the Device |
| 2211 | Current inside the device, No. 1 |
| 2212 | Current inside the device, No. 2 |
| 2213 | Overcurrent during Startup |
| 2214 | Overcurrent during Slowdown |
| 2220 | Continuous Overcurrent |
| 2221 | Continuous Overcurrent No. 1 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Code Value [Hex] | Meaning |
|---|---|
| 2222 | Continuous Overcurrent No. 2 |
| 2230 | Short Circuit/Short to earth |
| 2240 | Short to earth |
| 2250 | Short Circuit |
| 2300 | Current, Device Output Side |
| 2310 | Continuous Overcurrent |
| 2311 | Continuous Overcurrent, No. 1 |
| 2312 | Continuous Overcurrent, No. 2 |
| 2320 | Short Circuit/Short to Earth |
| 2330 | Short to Earth |
| 2331 | Short to Earth in Phase U |
| 2332 | Short to Earth in Phase V |
| 2333 | Short to Earth in Phase W |
| 2340 | Short Circuit |
| 2341 | Short Circuit in Phases U-V |
| 2342 | Short Circuit in Phases V-W |
| 2343 | Short Circuit in Phases W-U |
| | |
| **3000** | **Voltage** |
| 3100 | Mains Voltage |
| 3110 | Mains overvoltage |
| 3111 | Mains overvoltage in phase L1 |
| 3112 | Mains overvoltage in phase L2 |
| 3113 | Mains overvoltage in phase L3 |
| 3120 | Mains undervoltage |
| 3121 | Mains undervoltage in phase L1 |
| 3122 | Mains undervoltage in phase L2 |
| 3123 | Mains undervoltage in phase L3 |
| 3130 | Phase Failure |
| 3131 | Failure of Phase L1 |
| 3132 | Failure of Phase L2 |
| 3133 | Failure of Phase L3 |
| 3134 | Phase Sequence |
| 3140 | Mains Frequency |
| 3141 | Mains Frequency too high |
| 3142 | Mains Frequency too low |
| 3200 | Voltage inside the Device |
| 3210 | Overvoltage inside the device |
| 3211 | Overvoltage No. 1 |
| 3212 | Overvoltage No. 2 |
| 3220 | Undervoltage inside the Device |
| 3221 | Undervoltage No. 1 |
| 3222 | Undervoltage No. 2 |
| 3230 | Charging Error |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Code Value [Hex] | Meaning |
|---|---|
| 3300 | Output Voltage |
| 3310 | Output Overvoltage |
| 3311 | Output overvoltage in Phase U |
| 3312 | Output overvoltage in Phase V |
| 3313 | Output overvoltage in Phase W |
| 3320 | Armature Circuit |
| 3321 | Armature Circuit Discontinuity |
| 3330 | Field Circuit |
| 3331 | Field Circuit Discontinuity |
| | |
| **4000** | **Temperature** |
| 4100 | Ambient Temperature |
| 4110 | Excess Ambient Temperature |
| 4120 | Inadequate Ambient Temperature |
| 4130 | Ingoing Ambient Temperature |
| 4140 | Outgoing Air Temperature |
| 4200 | Device Temperature |
| 4210 | Excess Device Temperature |
| 4220 | Inadequate Device Temperature |
| 4300 | Drive Temperature |
| 4310 | Excess Drive Temperature |
| 4320 | Inadequate Drive Temperature |
| 4400 | Power Supply Temperature |
| 4410 | Excess Supply Temperature |
| 4420 | Inadequate Supply Temperature |
| | |
| **5000** | **Device Hardware** |
| 5100 | Power Supply |
| 5110 | Low Voltage Power Supply |
| 5111 | ± 15V Power Supply |
| 5112 | +24V Power Supply |
| 5113 | +5V Power Supply |
| 5120 | DC Link Power Supply |
| 5200 | Control |
| 5210 | Measurement Circuit |
| 5220 | Computing Circuit |
| 5300 | Operator Control Circuit |
| 5400 | Power Section |
| 5410 | Output Stages |
| 5420 | Chopper |
| 5430 | Input Stages |
| | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Code Value [Hex] | Meaning |
|---|---|
| **6000** | **Device Software** |
| 6010 | Software Reset (Watchdog) |
| 6100 | Internal Software |
| 6200 | User Software |
| 6300 | Date Set |
| 6301 | Data Set No.1 |
| 6302 … 630E | from 2 to 14 accordingly |
| 630F | Data Set No.15 |
| 6310 | Parameter Loss |
| 6320 | Parameter Error |
| | |
| **7000** | **Additional Modules** |
| 7100 | Power |
| 7110 | Brake Chopper |
| 7111 | Brake Chopper Failure |
| 7112 | Brake Chopper overcurrent |
| 7113 | Brake Chopper Wiring |
| 7120 | Motor |
| 7121 | Motor Blocked |
| 7200 | Measurement Circuit |
| 7300 | Sensor |
| 7301 | Tacho defective |
| 7302 | Wrong Tacho Polarity |
| 7303 | Resolver 1 defective |
| 7304 | Resolver 2 defective |
| 7305 | Incremental Encoder 1 Defective |
| 7306 | Incremental Encoder 2 Defective |
| 7307 | Incremental Encoder 3 Defective |
| 7310 | Speed |
| 7320 | Position |
| 7400 | Computing Circuit |
| 7410 | Velocity Ramp Generation |
| 7420 | Trajectory Generation Error |
| 7421 | Invalid Parameters |
| 7422 | Trajectory Generator Precalculation |
| 7423 | Trajectory Interpolation |
| 7500 | Communication |
| 7510 | Serial Interface No 1 |
| 7520 | Serial Interface No 2 |
| 7600 | Data Memory |
| | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Code Value [Hex] | Meaning |
|---|---|
| **8000** | **Monitoring** |
| 8100 | Communication |
| 8110 | Operational Data Monitoring |
| 8111 | No SYNC |
| 8112 | Synchronisation Fault |
| 8113 | No COMMAND |
| 8114 | COMMAND outside window. |
| 8115 | Cannot transmit ACTUAL |
| 8116 | ACTUAL outside window. |
| 8120 | Host Monitoring |
| 8200 | Closed Loop Control |
| 8300 | Torque Controller |
| 8302 | Torque Limiting |
| 8311 | Excess Torque |
| 8312 | Heavy Starting |
| 8313 | Standstill Torque |
| 8321 | Inadequate Torque |
| 8331 | Torque Breakage |
| 8400 | Speed Controller |
| 8401 | Velocity Following Error |
| 8402 | Velocity Limiting |
| 8500 | Position Controller |
| 8501 | Position Following Error |
| 8502 | Position Limiting |
| 8600 | Positioning Controller |
| 8611 | Following Error |
| 8612 | Reference Limit |
| 8700 | Synchro Controller |
| 8800 | Winding Controller |
| | |
| **9000** | **External Malfunction** |
| | |
| **F000** | **Additional Functions** |
| F001 | Deceleration |
| F002 | Inadequate Synchronization |
| F003 | Lifting Mechanism |
| F004 | Open LOOP Control |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-29.11 Abbreviated (8-bit) Fault and Warning Codes**

| Code Value | Meaning |
|---|---|
| 0 | NO FAULT |
| 10 | TEST |
| 11 | FORCE TRIP |
| 12 | E-STOP |
| 13 | RESTART RETRIES |
| 14 | SHORT CIRCUIT |
| 15 | AUXILLIARY I/O FAULT |
| 16 | NETWORK POWER |
| 20 | CURRENT TRIP |
| 21 | THERMAL OVERLOAD |
| 22 | PHASE LOSS |
| 23 | PHASE LOSS - A |
| 24 | PHASE LOSS - B |
| 25 | PHASE LOSS - C |
| 26 | PHASE IMBALANCE |
| 27 | GROUND FAULT |
| 28 | JAM |
| 29 | UNDERLOAD |
| 30 | MOTOR PLUG ATTEMPT |
| 31 | STALL |
| 32 | PHASE TO PHASE SHORT |
| 33 | PHASE TO GROUND SHORT |
| 40 | CONTROL VOLTAGE |
| 41 | CONTROL UNDERVOLTAGE |
| 42 | CONTROL OVERVOLTAGE |
| 43 | FREQUENCY |
| 44 | SPEED SWITCH |
| 50 | MAIN VOLTAGE |
| 51 | UNDERVOLTAGE |
| 52 | OVERVOLTAGE |
| 53 | IMBALANCE |
| 54 | PHASE REVERSAL |
| 55 | FREQUENCY |
| 56 | OVERPOWER |
| 57 | UNDERPOWER |
| 58 | VAR |
| 60 | HARDWARE |
| 61 | ILLEGAL FLC SETTING |
| 62 | MEMORY FAULT |
| 63 | HARDWARE LINK FAULT |
| 64 | NO DEVICE POWER |
| 65 | WATCHDOG TIMEOUT |
| 66 | INTERNAL COMPONENT OVER TEMPERATURE |
| 67 | TEMPERATURE SENSOR FAULT |
| 68 | TEMPERATURE SENSOR LOSS |
| 69 | INTERNAL COMMUNICATION/CONTROL LOST |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Code Value | Meaning |
|---|---|
| 70 | MISC |
| 71 | FAIL TO CLOSE |
| 72 | FAIL TO OPEN |
| 73 | STARTS/HOUR EXCEEDED |
| 74 | LOW POWER FACTOR |
| 75 | STATOR OVERTEMP |
| 76 | BEARING OVERTEMP |
| 77 | INCOMPLETE SEQUENCE |
| 78 | CONTACTOR #2, FAILED TO CLOSE |
| 79 | CONTACTOR #2, FAILED TO OPEN |
| 80 | CONTACTOR #3, FAILED TO CLOSE |
| 81 | CONTACTOR #3, FAILED TO OPEN |
| 82 | STATOR WINDING SHORT |
| 83 | MOTOR BEARING FAULT |
| 84 | EXCESSIVE VIBRATION |
| 85 | UNSTABLE DRIVEN LOAD |
| 86 | LOOSE MOTOR ROTOR BAR |
| 87 | CENTRIFUGAL UNBALANCE |
| 88 | ROTATIONAL UNBALANCE |
| 89 | MOTOR/LOAD MISALIGNMENT |
| 90 | MOTOR OPERATING OUTSIDE LOAD PROFILE |
| 91 | HIGH HARMONIC DISTORTION |
| 92 | MOTORING (MOTOR=GENERATOR OR NEGATIVE CURRENTS) |
| * Note Fault Codes 101 to 255 are Vendor Specific Codes | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-29.7 Object-specific General and Extended Status Codes

**Table 5A-29.12 Object-specific General and Extended Status Codes**

| General Status Codes (in hex) | 8-bit/16-bit Associated Extended Status Codes | Status Name | Description of Status |
|---|---|---|---|
| 00 -CF | | General Codes | Defined by CIP specification in Appendix B |
| | 00 - FE | | Reserved Extended Status Codes |
| | F0 - FE | | Vendor specific Extended Codes |
| | FF | | Used with all General Codes when required and no other Extended Code is assigned |
| D0 | | DRIVECOM Warning Code | The Extended Status Code is a DRIVECOM (16-bit) Warning code |
| | See Table 5A-29.10 | | DRIVECOM (16-bit) Warning Code |
| D1 | | DRIVECOM Fault Code | The Extended Status Code is a DRIVECOM (16-bit) Fault code |
| | See Table 5A-29.10 | | DRIVECOM (16-bit) Fault Code |
| D2 | | Abbreviated Warning Code | The Extended Status Code is an Abbreviated (8-bit) Warning Code |
| | See Table 5A-29.11 | | Abbreviated (8-bit) Warning Code |
| D3 | | Abbreviated Fault Code | The Extended Status Code is an Abbreviated (8-bit) Warning Code |
| | See Table 5A-29.11 | | Abbreviated (8-bit) Warning Code |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-30 AC/DC Drive Object

## Class Code: 2A hex

This object models the functions specific to an AC or DC Drive. e.g. speed ramp, torque control etc.

## 5A-30.1 Class Attributes

**Table 5A-30.1 AC/DC Drive Object Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |

## 5A-30.2 Instance Attributes

**Table 5A-30.2 AC/DC Drive Object Instance Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|---|
| 1 | Optional | Get | | NumAttr | USINT | Number of Attributes supported |
| 2 | Optional | Get | | Attributes | Array of USINT | List of Attributes supported |
| 3 | Optional | Get | | AtReference | BOOL | 1 = Drive actual at reference (speed or torque reference) based on mode |
| 4 | Required | Set/ Get | | NetRef | BOOL | Requests torque or speed reference to be local or from the network.<br>0 = Set Reference not DN Control<br>1 = Set Reference at DN Control<br>Note that the actual status of torque or speed reference is reflected in attribute 29, RefFromNet. |
| 5 | Optional | Set/ Get | | NetProc | BOOL | Requests process control reference to be local or from the network.<br>0 = Set Process not DN Control<br>1 = Set Process at DN Control<br>Note that the actual status of the process control reference is reflected in attribute 30, ProcFromNet. |
| 6 | Required | Set/ Get | | DriveMode | USINT | 0 = Vendor specific mode<br>1 = Open loop speed (Frequency)<br>2 = Closed loop speed control<br>3 = Torque control<br>4 = Process control (e.g. PI)<br>5 = Position control |
| 7 | Required | Get | | SpeedActual | INT | Actual drive speed (best approximation)<br>Units: RPM / $2^{SpeedScale}$<br>where SpeedScale is attribute 22 |
| 8 | Required | Set/ Get | | SpeedRef | INT | Speed reference<br>Units: RPM / $2^{SpeedScale}$<br>where SpeedScale is attribute 22 |
| 9 | Optional | Get | | CurrentActual | INT | Actual motor phase current<br>Units: 100ma / $2^{CurrentScale}$<br>where CurrentScale is attribute 23 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## AC/DC Drive Object, Class Code: $2A_{Hex}$

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|---|
| 10 | Optional | Set/ Get | | CurrentLimit | INT | Motor phase current limit<br>Units: 100ma/ $2^{CurrentScale}$<br>where CurrentScale is attribute 23 |
| 11 | Optional | Get | | TorqueActual | INT | Actual torque<br>Units: Nm / $2^{TorqueScale}$<br>where TorqueScale is attribute 24 |
| 12 | Optional | Set/ Get | | TorqueRef | INT | Torque reference<br>Units: Nm / $2^{TorqueScale}$<br>where TorqueScale is attribute 24 |
| 13 | Optional | Get | | ProcessActual | INT | Actual process control value<br>Units: % / $2^{ProcessScale}$<br>where ProcessScale is attribute 25 |
| 14 | Optional | Set/ Get | | ProcessRef | INT | Process control reference set point<br>Units: % / $2^{ProcessScale}$<br>where ProcessScale is attribute 25 |
| 15 | Optional | Get | | PowerActual | INT | Actual output power<br>Units: Watts / $2^{PowerScale}$<br>where PowerScale is attribute 26 |
| 16 | Optional | Get | | InputVoltage | INT | Input Voltage<br>Units: Volts / $2^{VoltageScale}$<br>where VoltageScale is attribute 27 |
| 17 | Optional | Get | | OutputVoltage | INT | Output Voltage<br>Units: Volts / $2^{VoltageScale}$<br>where VoltageScale is attribute 27 |
| 18 | Optional | Set/ Get | | AccelTime | UINT | Acceleration time<br>Time from 0 to HighSpdLimit<br>Units: ms / $2^{TimeScale}$<br>where TimeScale is attribute 28<br>Acceleration time selection for negative direction is vendor specific. |
| 19 | Optional | Set/ Get | | DecelTime | UINT | Deceleration time<br>Time from 0 to HighSpdLimit<br>Units: ms / $2^{TimeScale}$<br>where TimeScale is attribute 28<br>Deceleration time selection for negative direction is vendor specific. |
| 20 | Optional | Set/ Get | | LowSpdLimit | UINT | Minimum speed limit<br>Units: RPM / $2^{SpeedScale}$<br>where SpeedScale is attribute 22 |
| 21 | Optional | Set/ Get | | HighSpdLimit | UINT | Maximum speed limit<br>Units: RPM / $2^{SpeedScale}$<br>where SpeedScale is attribute 22 |
| 22 | Optional* | Set/ Get | | SpeedScale | SINT | Speed scaling factor. Scaling is accomplished as follows:<br>ScaledSpeed = RPM / $2^{SpeedScale}$<br>Range: -128 .. 127 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|---|
| 23 | Optional* | Set/ Get | | CurrentScale | SINT | Current scaling factor. Scaling is accomplished as follows:<br>ScaledCurrent = A / $2^{CurrentScale}$<br>Range: -128 .. 127 |
| 24 | Optional* | Set/ Get | | TorqueScale | SINT | Torque scaling factor. Scaling is accomplished as follows:<br>ScaledTorque = Nm / $2^{TorqueScale}$<br>Range: -128 .. 127 |
| 25 | Optional* | Set/ Get | | ProcessScale | SINT | Process scaling factor. Scaling is accomplished as follows:<br>ScaledProcess = % / $2^{ProcessScale}$<br>Range: -128 .. 127 |
| 26 | Optional* | Set/ Get | | PowerScale | SINT | Power scaling factor. Scaling is accomplished as follows:<br>ScaledPower = W / $2^{PowerScale}$<br>Range: -128 .. 127 |
| 27 | Optional* | Set /Get | | VoltageScale | SINT | Voltage scaling factor. Scaling is accomplished as follows:<br>ScaledVoltage = V / $2^{VoltageScale}$<br>Range: -128 .. 127 |
| 28 | Optional* | Set/ Get | | TimeScale | SINT | Time scaling factor. Scaling is accomplished as follows:<br>ScaledTime = ms / $2^{TimeScale}$<br>Range: -128 .. 127 |
| 29 | Optional | Get | | RefFromNet | BOOL | Status of torque/speed reference<br>0=Local torque/speed reference<br>1=Network torque/speed reference |
| 30 | Optional | Get | | ProcFromNet | BOOL | Status of process control reference<br>0=Local process reference<br>1=Network process reference |
| 31 | Optional | Set/ Get | | FieldIorV | BOOL | Selects Field Voltage or Field Current control for a DC Drive.<br>0=Voltage Control (Open Loop)<br>1=Current Control<br>(Magnetizing field for DC drive) |
| 32 | Optional | Set/ Get | | FieldVoltRatio | UINT | For voltage control of a DC Drive |
| 33 | Optional | Set/ Get | | FieldCurSetPt | UINT | DC Drive Field Current set point.<br>Units: Amps / $2^{CurrentScale}$<br>where CurrentScale is attribute 23 |
| 34 | Optional | Set/ Get | | FieldWkEnable | BOOL | Enables/Disables field weakening for a DC Drive<br>0=Disabled (DC Drive in current control)<br>1=Enabled |
| 35 | Optional | Get | | FieldCurActual | INT | Actual Field Current for a DC Drive.<br>Units: Amps / $2^{CurrentScale}$<br>where CurrentScale is attribute 23 |
| 36 | Optional | Set/ Get | | FieldMinCur | INT | Minimum Field Current for a DC Drive.<br>Units: Amps / $2^{CurrentScale}$<br>where CurrentScale is attribute 23 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|---|
| 37 | Optional | Set | NV | Process Data Units | ENGUNITS | Applies to:<br>ProcessActual<br>ProcessRef<br>Values specified in Appendix D. |
| 38 | Optional | Set | NV | Speed Control | BYTE | Set bits are used to request speed commands. See Semantics section. |
| 39 | Optional | Get | V | Speed Status | BYTE | Set bits indicate the current speed status.<br>see Semantics section. |
| 40 | Optional | Set | NV | Speed Trip Time | UINT | Time for SpeedActual beyond vendor specific hysteresis band. See Behavior section.<br>Units: ms / $2^{TimeScale}$<br>where TimeScale is attribute 28 |
| 41 | Optional | Get | NV | Max Rated Speed | INT | Vendor Specific maximum rating.<br>Units: RPM / $2^{MRSScale}$<br>where MRSScale is attribute 46 |
| 42 | Optional | Set | NV | Max Rated Speed Scale | SINT | Speed Scaling for the Maximum Rating. Scaling is accomplished as follows:<br>ScaledSpeed = RPM / $2^{MRSScale}$<br>Range: -128 .. 127 |
| 43 | Optional | Set | NV | SpeedStandby | INT | Speed setting for Standby. See Behavior section.<br>Units: RPM / $2^{SpeedScale}$<br>where SpeedScale is attribute 22 |
| 44 | Optional | Set | NV | SpeedActual Data Units | ENGUNITS | Allows a change from RPM<br>RPM [default]<br>Applies to:<br>SpeedActual<br>SpeedStandby<br>Values specified in Appendix D |
| 45 | Optional | Set | NV | SpeedRef Data Units | ENGUNITS | Allows a change from RPM<br>RPM [default]<br>Applies to:<br>SpeedRef<br>Max Rated Speed<br>LowSpdLimit<br>HighSpdLimit<br>Values specified in Appendix D. |
| 46 | Optional | Get | NV | Drive On Hours | DINT | Number of hours SpeedActual > 0<br>Value shall not "roll over" at terminal count |

* See section 5A-30.5.2 for scaling example.

The basic units used by the AC/DC Drive Object are:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-30.3 Basic Units**

| Physical Quantity | Units | |
|---|---|---|
| Speed: | RPM | Revolutions Per Minute |
| Current: | 100ma | 100 milliAmps |
| Torque: | Nm | Newton Meters |
| Process control: | % | Percent |
| Power: | W | Watts |
| Voltage: | V | Volts |
| Time: | ms | Milliseconds |

### 5A-30.2.1 Semantics

#### 5A-30.2.1.1 Speed Control and Speed Status

The *Speed Control* attribute is used to request various commands for the speed objective of the AC/DC Drive object. The value of the *Speed Status* attribute indicates the current speed status. See Behavior section for more information. Setting a bit to TRUE effects the request for the respective speed. A request to set the *Speed Control* attribute to an unsupported bit pattern (i.e., representing a command that the object does not support) shall yield an Invalid Attribute Value error response.

**Table 5A-30.4 *Speed Control* Attribute and *Speed Status* Attribute Bit Map**

| Bit | Speed Control | Speed Status | Status Description |
|---|---|---|---|
| 0 | Run Request | Running | On and SpeedActual > 0 |
| 1 | Idle Request | At Idle | Zero current |
| 2 | Standby Request | At Standby Speed | SpeedActual = SpeedStandby |
| 3 | Coast Request | Coasting | Zero Torque |
| 4 | n.a. * | Stopped | SpeedActual = 0 |
| 5 | n.a. * | Accelerating | SpeedActual is increasing |
| 6 | n.a. * | At Reference ** | SpeedActual = SpeedRef |
| 7 | n.a. * | Decelerating | SpeedActual is decreasing |

* value shall be ignored
** corresponds to the AtReference attribute (ID 3).

When these attributes (Speed Control and Speed Status) are supported by a device with a Control Supervisor object, *Speed Control* Bit 0 corresponds to *Run1* (Control Supervisor instance attribute ID 3) and *Speed Status* Bit 0 corresponds to Running1 (Control Supervisor instance attribute ID 7).

Some implementations may support only some values of *Speed Status*. For example, At Idle and Stopped may always be set together (i.e., no difference between these two status conditions). A detailed description of the actual behavior corresponding to these speed conditions is vendor specific.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-30.3 Common Services

**Table 5A-30.5 AC/DC Drive Object Common Services**

| Service | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| **Code** | **Class** | **Instance** | | |
| 0E hex | Conditional* | Required | Get_Attributes_Single | Returns the contents of the specified attribute. |
| 10 hex | n/a | Required | Set_Attributes_Single | Modifies an attribute value. |
| 15 hex | n/a | Optional | Restore | Restores attribute values from a storage location accessible by the Save service. This service is only available if the drive is in the Ready or NOT_READY states. |
| 16 hex | n/a | Optional | Save | Saves all attribute values to a location accessible by the Restore service. |
| See Appendix A for definition of these services | | | | |

## 5A-30.4 Object-specific Services

The AC/DC Drive object provides no object specific services.

## 5A-30.5 Behavior

The following drawing represents the signal flow in an AC drive whose output is an inverter frequency command. Signal flow is controlled by the AC/DC Drive Instance attributes shown in **Bold** type. The process and speed references can be directed to originate from network commands or from references internal to the drive or supplied to the drive through terminal blocks or keypads. Network control of these alternative reference sources will be vendor specific. The mode of operation can also be controlled from the network with the mode attribute AC/DC Drive object. Note that not all modes will be supported by all drives.

**Figure 5A-30.1 AC/DC Drive Object Behavior - Frequency Inverter**

The following drawing represents the signal flow for a AC or DC torque or current controlled drive. These drives can operate in Mode 3, allowing a network torque reference to directly control the drive.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Each drive vendor can implement a process regulator and position regulator to feed either the speed regulator or the torque command directly. Broken lines indicate these alternative signal paths.

**Figure 5A-30.2 AC/DC Drive Object Behavior – Torque or Current Controlled**

As illustrated below, **AccelTime** (attribute 18) is defined as the time in seconds it would take the drive to accelerate from zero to **HighSpdLimit** (attribute 21), the maximum speed at which the drive is allowed to operate.

Similarly, **DecelTime** (attribute 19) is the time to slow down from **HighSpdLimit** (attribute 21) to zero speed.

There will be no guarantee that the drive will operate at these rates in actual applications, since current limit may increase the acceleration time, and lack of sufficient braking capability may increase the stopping time.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5A-30.3 Accel/Decel Time**

When **DriveMode** (attribute 6) is set to "1" or "2", **AtReference** (attribute 3) is "1" when **SpeedActual** (attribute 7) is at its prescribed speed reference plus or minus a vendor specific hysteresis band. The vendor specific hysteresis band may be fixed, programmable, or zero.

**Figure 5A-30.4 Vendor Specific Hysteresis Band – Speed Reference**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

When **DriveMode** (attribute 6) is set to "3", **AtReference** (attribute 3) is "1" when **TorqueActual** (attribute 11) is at its prescribed torque reference plus or minus a vendor specific hysteresis band. The vendor specific hysteresis band may be fixed, programmable, or zero.

**Figure 5A-30.5 Vendor Specific Hysteresis Band – Torque Reference**

### 5A-30.5.1 Enhanced Speed Control Behavior

The *Speed Control* attribute is used to request different speed objectives for the AC/DC Drive object. With no bits set to True, the objective for the object is to obtain the speed specified by the value of the *SpeedRef* attribute. Other Speed Control bits are used to request alternative speed objectives, including: idle, standby and coast.

For DriveMode=1,2, the Standby alternative speed objective alters the source of the speed setpoint (input to Accel/Decel Circuit) from *SpeedRef* (attribute 8) to *SpeedStandby* (attribute 43).

Irrespective of DriveMode; idle and coast affect the appropriate action by overriding the Accel/Decel Circuit. SpeedRef and SpeedStandby are ignored by the Accel/Decel Circuit while idle or coasting.

The *Speed Trip Time* attribute specifies a time for which the value of *SpeedActual* may exceed the vendor specific hysteresis band before an Speed Trip Condition exists. The particular actions and/or behavior modifications associated with an Speed Trip Condition shall be specified by the Device Profile.

### 5A-30.5.2 Scaling of attribute values

As part of the AC/DC Drive Object definition, engineering units are defined for each physical quantity, e.g. RPM for Velocity, Nm for Torque etc. To maximize the resolution capable or necessary on some devices or applications, these values can be normalized using a binary scale factor before transmission on the bus. A separate scaling factor is specified for each physical quantity. Normally, scaling factors will be set up once during initialization according to the range of values to be used in the application.

Scaling Factors allow the representation of physical units on the bus to obtain an acceptable resolution and dynamic range for all applications.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Example**: Configuration of a DC Drive to operate with RPM resolution of 0.125 RPM

SpeedRef (AC/DC Drive Object, Attribute ID 8) = 4567

SpeedScale (AC/DC Drive Object, Attribute ID 22) = 3

=> Actual Commanded Speed = SpeedRef / $2^{\text{SpeedScale}}$ = 4567 / $2^{3}$ = 570.875 RPM

Input from Drive to bus:

Actual Drive Operating Speed = 789.5 RPM

SpeedScale (AC/DC Drive Object, Attribute ID 22) = 3

⇒ SpeedActual (AC/DC Drive Object, Attribute ID 7) =

Actual Operating Speed × $2^{\text{SpeedScale}}$ = 789.5 × $2^{3}$ = 6316.

In cases where the applicable scaling factor attribute is non-zero, the units are:

Engineering unit / $2^{\text{Scale Factor Attribute}}$

In the above example, therefore, the units are 0.125 RPM.

Drives that do not support scale factors (Attributes 22 ~ 28), should return an "Attribute Not Supported" error. In this case, the default scaling is presumed, and all physical quantities will then default to engineering units. For example, speed values will then default to units of RPM. Drives that do not use RPM as an internal measure should calculate an RPM value based on motor parameters. For example, AC VFD's may calculate an RPM/Hz constant using motor rated frequency and motor base speed from theMotor Data Object. An example of this type of implementation (including scaling factors) for the commanded VFD's frequency may be:

Frequency Command (Hz) = (SpeedRef x RatedFreq) / ($2^{\text{SpeedScale}}$ x BaseSpeed)

and an example implementation for the reported operating speed as a function of the output frequency may be:

SpeedActual = Int[ (Output Frequency (Hz) x BaseSpeed x $2^{\text{SpeedScale}}$) / RatedFreq ]

Where:

Frequency Command = the VFD's resultant frequency command, the resolution of which is vendor-dependent.

SpeedRef = AC/DC Drive Object Instance Attribute ID 8.

RatedFreq = Motor Data Object, AC Motor Instance Attribute ID 9.

SpeedScale = AC/DC Drive Object Instance Attribute ID 22.

BaseSpeed = Motor Data Object, AC Motor Instance Attribute ID 15.

SpeedActual = AC/DC Drive Object Instance Attribute ID 7.

Int = A vendor-specific integer typecast function (such as rounding or truncation).

Output Frequency = the VFD's reported present operating frequency, the resolution of which is vendor-dependent.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-31 Acknowledge Handler Object

## Class Code: 2B Hex

The Acknowledge Handler Object is used to manage the reception of message acknowledgments. This object communicates with a message producing Application Object within a device. The Acknowledge Handler Object notifies the producing application of acknowledge reception, acknowledge timeouts, and production retry limit.

## 5A-31.1 Class Attributes

**Table 5A-31.1 Acknowledge Handler Object Class Attributes**

| Number | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |

## 5A-31.2 Class Services

- Get_Attribute_Single
- Create
- Delete

**Table 5A-31.2 Acknowledge Handler Object Common Services - Class Level**

| Need in Implementation | Name | Service Code | Description of Service |
|---|---|---|---|
| Optional | Get_Attribute_Single | 0E hex | Used to read an Acknowledge Handler Object attribute value. |
| Optional | Create | 08 hex | Used to create an Acknowledge Handler Object. |
| Optional | Delete | 09 hex | Used to delete all dynamically created Acknowledge Handler Object instances. |
| See Appendix A for definitions of these services | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-31.3 Instance Attributes

**Table 5A-31.3 Acknowledge Handler Object Instance Attributes**

| Attr ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Set | Acknowledge Timer | UINT | Time to wait for acknowledge before resending | Range 1-65,535 ms (0 invalid) default = 16 |
| 2 | Required (Get) Optional (Set) | Get/Set | Retry Limit | USINT | Number of Ack Timeouts to wait before informing the producing application of a RetryLimit_Reached event. | Range 0-255 default = 1 |
| 3 | Required | Set (Inactive) Get (Active) | COS Producing Connection Instance | UINT | Connection Instance which contains the path of the producing I/O application object which will be notified of Ack Handler events. | Connection Instance ID |
| 4 | Optional | Get | Ack List Size | BYTE | Maximum number of members in Ack List | 0 = Dynamic >0 = Max number of members |
| 5 | Optional | Get | Ack List | BYTE Array of UINT | List of active connection instances which are receiving Acks. | Number of members followed by list of: Connection Instance ID |
| 6 | Optional | Get | Data with Ack Path List Size | BYTE | Maximum number of members in Data with Ack Path List | 0 = Dynamic >0 = Max number of members |
| 7 | Optional | Get | Data with Ack Path List | BYTE Array of UINT USINT Padded EPATH | List of connection instance/consuming application object pairs. This attribute is used to forward data received with acknowledgment. | Number of members followed by list of: Connection Instance ID/CIP path length/CIP path |

- If the specified value for the Acknowledge Timer attribute is not equal to an increment of the available clock resolution, then the value is rounded up to the next serviceable value. For example: a Set_Attribute_Single request is received specifying the value 5 for the Acknowledge Timer attribute and the product provides a 10 millisecond resolution on timers. In this case the product would load the value 10 into the Acknowledge Timer attribute.
- The value that is actually loaded into the Acknowledge Timer attribute is reported in the Service Data Field of a Set_Attribute_Single response message associated with a request to modify this attribute.
- A successful set attribute to the Retry Limit attribute will reset the Retry Counter.
- The Ack List attribute is updated when an associated connection transitions between configuring, established, timed-out, and non-existent. Refer to the State Event Matrix for details.
- If the Acknowledge Handler Object is active (at least one member in the Ack List), the COS Producing Connection Instance attribute access is get only.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- The default value loaded into the Acknowledge Timer attribute at time of instantiation is 16 ms.
- The default value loaded into the Retry Limit attribute at time of instantiation is 1.

## 5A-31.4 Instance Services

The Acknowledge Handler Object Instance supports the following Common Services:

- Get_Attribute_Single
- Set_Attribute_Single
- Delete

**Table 5A-31.4 Acknowledge Handler Object Common Services – Instance Level**

| Need in Impl | Name | Service Code | Description of Service |
|---|---|---|---|
| Required | Get_Attribute_Single | 0E hex | Used to read an Acknowledge Handler Object attribute value. |
| Required | Set_Attribute_Single | 10 hex | Used to modify an Acknowledge Handler Object attribute value. |
| Optional | Delete | 09 hex | Used to delete an Acknowledge Handler Object. |
| See Appendix A for definition of these services | | | |

## 5A-31.5 Object-specific Services

The Acknowledge Handler Object Instance also supports the following Object Class Specific Services:

- Add_AckData_Path
- Remove_AckData_Path

**Table 5A-31.5 Acknowledge Handler Object Object-specific Services**

| Need in Impl | Name | Service Code | Description of Service | Service Request Parameters |
|---|---|---|---|---|
| Optional | Add_AckData_Path | 4B hex | Adds a path for data with acknowledgment for a connected consumer. | Connection Instance<br>CIP Path Length<br>CIP Path |
| Optional | Remove_AckData_Path | 4C hex | Removes a path with acknowledge path for the given connected consumer. | Connection Instance |

These services are used to add and remove paths for each of the connected acknowledge consumers when data is sent with the acknowledgment.

## 5A-31.6 Behavior and Configuration of Acknowledged Data Production

The following rules are used to configure and determine the behavior of an acknowledged Change of State or Cyclic I/O connection using the Acknowledge Handler Object. In the following examples, COS Producer is used to reference the device producing change of state or cyclic data and consuming an acknowledgment (client). COS Consumer is used to reference the device consuming the change of state or cyclic data and producing an acknowledgment (server).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-31.6.1 Acknowledged Data Production

1. The COS Producers consumed connection path must be set to an available Acknowledge Handler Object. The path must consist of Class and Instance. If an Acknowledge Handler Object is not available, use the Acknowledge Handler Class Create service to obtain a new one.
2. The COS Producers producing I/O application informs the Acknowledge Handler Object of new data production (Data Sent event message) or data production retries (Data Resent event message).
3. The COS Producers acknowledgment reception is done by the Acknowledge Handler Object. The Acknowledge Handler object informs the Producing I/O application when one or more acknowledges have not been received within the acknowledge timeout (using the Ack List and Ack Timeout attributes).
4. The acknowledge message requires no data. The COS producing device's Acknowledge Handler object should consider valid message reception as an acknowledgment. However, a change of state or cyclic producing device may be configured to consume data along with the acknowledgment. In this case the data is forwarded to the application object in the 'Data with Ack Path List' attribute, based on the connection which received the data.
5. The COS Consumer acknowledge producing application should be configured to send either a zero length message or valid response (output) message when a valid input message is consumed.
6. An acknowledge timer is started each time production occurs. The Acknowledge Handler object is notified of this event by a Data Sent or Data Resent event message from the producing application.
7. Expiration of the acknowledge timer causes an Acknowledge Timout message to be sent to the producing application object. That object must resend the last message if the Retry Limit has not been reached. It may also take an application specific action.
8. The retry count is incremented each time an Acknowledge Timeout message is sent to the producing application. When the retry limit has been reached, a Retry Limit Reached message is sent to the producing application object.
9. The retry count is cleared on each Data Sent message. A Data Resent message does *not* clear the retry counter.
10. The acknowledge timer value is configurable within the Acknowledge Handler object.
11. The number of retries is (optionally) configurable within the Acknowledge Handler object.

### 5A-31.6.2 Change of State Examples

#### 5A-31.6.2.1 Acknowledged Change of State (Using one connection object and one COS consumer)

For the producer:

1. Create a Connection Object.
2. The producer transportClass_trigger attribute is set to Class 2/3, Change-Of-State, Client (13h for Class 3) or Class2/3, Cyclic, Client (03h for Class 3).
3. The produced connection path is set to the producing application, which must support Change of State or Cyclic production.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

4. Create an Acknowledge Handler object. Configure the 'COS Producing Connection Instance' attribute to the instance of the connection object which specifies the producing I/O application object. The Ack List attribute will be updated with the connection instance of the I/O connection object which is consuming an acknowledge as that connection transitions to the established state. Optionally configure the Acknowledge Timer and Retry Count attributes.
5. The consumed connection path is set to the Acknowledge Handler object just created and configured for handling the acknowledge. The path contains Class and Instance only.
6. The consumed connection size is set to the size of the data expected to be delivered to the object specified in the consumed path of the Acknowledge Handler Object, or zero if no path is configured.
7. All other attributes are configured the same way as any other peer to peer connection.

For the consumer:

1. The consumer transportClass_trigger attribute is set to Class 2/3, Server (83h for Class 3).
2. The produced connection path is set to the consuming application (or some other application, this is product specific) for generating the acknowledge.
3. The produced connection size is set to the size of the data to be delivered to the object specified in the consumed path of the change of state producing device's Acknowledge Handler object, or zero if no path is configured.
4. All other attributes are configured the same way as any other peer to peer connection.

#### 5A-31.6.2.2 Acknowledged Change of State (Using multiple connection objects and COS consumers)

For the producer:

1. Create a Connection Object.
2. The producer transportClass_trigger attribute is set to Class 0, Change-Of-State, Client (10h) or Class 0, Cyclic, Client (00h).
3. The produced connection path is set to the producing application which must support Change of State or Cyclic production.
4. The consumed connection path and consumed connection size are not configured.
5. All other attributes are configured the same way as any other peer to peer connection.
6. Create a Connection Object for each COS consumer.
7. The consumer transportClass_trigger attribute is set to Class 0, Server (80h) for each consuming connection object.
8. Create an Acknowledge Handler object. Configure the 'COS Producing Connection Instance' attribute to the instance of the connection object which specifies the producing I/O application object. The Ack List attribute will be updated with the connection instance of each I/O connection object which is consuming an acknowledge as those connections transition to the established state. Optionally configure the Acknowledge Timer and Retry Count attributes.
9. The consumed connection path for each consuming connection object is set to the Acknowledge Handler object just created and configured for handling the acknowledge. The path contains Class and Instance only.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

10. The consumed connection sizes are set to the size of the data expected to be delivered to the consumed path set in the Acknowledge Handler Object, or zero if no path is configured.
11. All other attributes are configured the same way as any other peer to peer connection.

For each consumer:

1. The consumer transportClass_trigger attribute is set to Class 2/3, Server (83h for Class 3).
2. The produced connection path is set to the consuming application (or some other application, this is product specific) for generating the acknowledge.
3. The produced connection size is set to the size of the data to be delivered to the consumed path set in the change of state producing device's Acknowledge Handler object, or zero if no path is configured.
4. All other attributes are configured the same way as any other peer to peer connection.

Once the Connection Object(s) is (are) configured, an Apply Attributes service is sent to each configured Connection Object to transition the connection(s) to the Established state. During the apply, the connection object is 'delivered' to both the consuming and producing application objects for validation of the attribute information. At this time, the producing I/O application checks for a change of state configuration by examining the transportClass_trigger attribute. If the Production Trigger bits are set to change-of-state or cyclic, the application will configure itself for change of state or cyclic production. If change of state or cyclic production is not supported by the producing application object an error must be returned to the apply_attributes service.

### 5A-31.6.3 Use of Timers with Acknowledged Data Production

The following rules must be observed when sending acknowledged data.

New data not sent while the Inhibit Timer is active (running).

New data is sent when no acknowledge is pending, subject to rule # 1. (An acknowledge is pending after a send of new data or a retry of old data and until an Ack Timeout or Ack Received.

Retry of old data occurs at Ack Timeout if new data is not available or the Inhibit Timer is active.

Sending new data (or old data on transmission trigger timeout) starts the Ack Timer, Inhibit Timer, and the Transmission Trigger Timer. The Retry Counter is also cleared.

A retry of old data starts the Ack Timer.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5A-31.1 Typical Timing Relationships for Acknowledged Data Production**

**Figure 5A-31.2 Example of a COS System, 2 Acking Devices**

The following diagrams illustrate the message flow in a Change of State connection for both single and multi-consumer configurations.

**Figure 5A-31.3 Message Flow in COS Connection – One Connection Object, One Consumer**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5A-31.4 Message Flow in COS Connection –Multiple Consumers**

The following are the State Event Matrices for the Producing I/O application object and Acknowledge Handler object associated with a Change of State connection.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-31.6 SEM for Producing I/O Application Object**

| State ► / Events ▼ | Not Running | Running | Running with Acknowledgment | Prohibited |
|---|---|---|---|---|
| **Change of State Detected** | Ignore Event | Inform Link Producer to Send Data.<br>If Inhibit Time configured<br>Start Inhibit Timer<br>Transition to Produced | Inform Link Producer to Send Data.<br>Send DataSent event message to Acknowledge Handler object.<br>If Inhibit Time configured<br>Start Inhibit Timer transition to Prohibited. | Queue Event |
| **Acknowledge Received** | Not Applicable | Not Applicable | Ignore Event | If Ack_Active Set<br>If Inhibit Timer not running<br>Transition to Running with Acknowledgement<br>Else<br>Set Ack_Receive flag<br>Else ignore event |
| **Acknowledge Timeout** | Ignore Event | Not Applicable | Inform Link Producer to Send Data<br>Send Data_Resent event message to Acknowledge Handler Object. | Inform Link Producer to Send Data.<br>Send Data_Resent event message to Acknowledge Handler Object. |
| **Transmission Timer Expires** | Not Applicable | Inform Link Producer to Send Data. | Inform Link Producer to Send Data.<br>Send Data_Sent event message to Acknowledge Handler Object. | Inform Link Producer to Send LAST Data Sent.<br>Send Data_Sent event message to Acknowledge Handler Object. |
| **Retry Limit Reached** | Ignore Event | Not Applicable | Product Specific | Transition to Running with Acknowledgement. |
| **Inhibit timer Expires** | Ignore Event | Not Applicable | Not Applicable | If Ack_Active set<br>If Ack_Received<br>Transition to Running w/Ack<br>Clear Ack_Received flag<br>Else Ignore Event<br>Else Transition to Running |
| **Connection Deleted** | Not Applicable | Transition to Not Running | Transition to Not Running | Transition to Not Running |
| **Acknowledge Active** | Set Ack Active Flag | Transition to Running with Acknowledgement<br>Set Ack Active Flag | Ignore Event | Set Ack_Active Flag |
| **Acknowledge Inactive** | Reset Ack Active Flag | Ignore Event | Transition to Running<br>Reset Ack Active Flag | Reset Ack_Active Flag |
| **Connection Transition to Established** | If Ack Active<br>Transition to Running with Acknowledgment<br>If Ack Inactive<br>Transition to Running | Ignore Event | Ignore Event | Ingore Event |

Note: This is a partial SEM for a producing I/O application object. Only those states and events associated with data acknowledgement are defined. Other states and events will most likely be associated with a producing I/O application object.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-31.7 SEM for Acknowledge Handler Object**

| State ► / Events ▼ | Non-Existent | Inactive | Active |
|---|---|---|---|
| **Receive Acknowledge** | Not Applicable | Ignore Event | Clear Ack Flag<br>Forward any data to application object<br>If all Acknowledges received<br>Clear Ack Timer and Retry Counter<br>Send Acknowledge_Received event<br>message to producing application object |
| **Acknowlege Timer Expires** | Not Applicable | Ignore Event | Send Ack_Timeout event message to producing application object. |
| **Data Sent** | Not Applicable | Ignore Event | Set Ack Required Flag<br>Set Ack Timer<br>Clear Retry Counter |
| **Data Resent** | Not Applicable | Ignore Event | Set Ack Required Flag & Ack Timer<br>If Retry_Limit > 0<br>Increment Retry Counter<br>If Retry Counter = Retry_Limit<br>Send Rety_Limit_Reached event<br>message to producing application object |
| **Delete** | Not Applicable | Transition to Non-Existent | Transition to Non-Existent |
| **Create** | Transition to Inactive | Not Applicable | Not Applicable |
| **Apply Attributes** | Not Applicable | Verify new connection can be added to list. Pass this message to the consumed application object, if one is configured for this connection. | Verify new connection can be added to list. Pass this message to the consumed application object, if one is configured for this connection. |
| **Connection Transition to Established** | Not Applicable | Add this connection instance to the connection list (or internally flag as 'Acking') .<br>Pass this message to the consumed application object, if one is configured for this connection.<br>Send Acknowledge_Active event message to producing application.<br>Transition to Active. | Add this connection instance to the connection list. Pass this message to the consumed application object, if one is configured for this connection. |
| **Inactivity/Watchdog Timer Expires** | Not Applicable | Not Applicable | Internally flag this connection as 'Not Acking'. An acknowlegement will no longer be monitored for this connection, however it remains in the connection list.<br>Pass this event to the consumed application object, if one is configured for this connection.<br>If no 'Acking' connections in list, send Acknowledge_Inactive event to producing application and transition to Inactive. |
| **Connection Deleted** | Not Applicable | Ignore Event | Remove this connection instance from the connection list.<br>Pass this event to the consumed application object, if one is configure for this connection.<br>If no 'Acking' connections in list, send Acknowledge_Inactive event to producing application and transition to Inactive. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-32 Overload Object

## Class Code: 2C Hex

This object models all the functions specific to an AC motor overload protection device.

## 5A-32.1 Class Attributes

**Table 5A-32.1 Overload Object Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | |

## 5A-32.2 Instance Attributes

**Table 5A-32.2 Overload Object Instance Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|
| 1 | Optional | Get | NumAttr | USINT | Number of Attributes supported |
| 2 | Optional | Get | Attributes | Array of USINT | List of Attributes supported |
| 3 | Optional | Set | TripFLCSet | INT | Overload Full Load Current Setting<br>Units : 100ma / $2^{CurrentScale}$<br>where CurrentScale is attribute 12 |
| 4 | Optional | Set | TripClass | USINT | Trip Class Setting<br>0 to 200 |
| 5 | Optional | Get | AvgCurrent | INT | Average of the three phase current.<br>Units: 100ma / $2^{CurrentScale}$<br>where CurrentScale is attribute 12 |
| 6 | Optional | Get | %PhImbal | USINT | % Phase Imbalance |
| 7 | Optional | Get | %Thermal | USINT | % Thermal Capacity |
| 8 | Optional | Get | CurrentLl | INT | Actual motor phase current L1<br>Units: 100ma / $2^{CurrentScale}$<br>where CurrentScale is attribute 12 |
| 9 | Optional | Get | CurrentL2 | INT | Actual motor phase current L2<br>Units: 100ma / $2^{CurrentScale}$<br>where CurrentScale is attribute 12 |
| 10 | Optional | Get | CurrentL3 | INT | Actual motor phase current L3<br>Units: 100ma / $2^{CurrentScale}$<br>where CurrentScale is attribute 12 |
| 11 | Optional | Get | GroundCurrent | INT | Ground Current<br>Units: 100ma / $2^{CurrentScale}$<br>where CurrentScale is attribute 12 |
| 12 | Optional | Set | CurrentScale | SINT | Current scaling factor. Scaling is accomplished as follows:<br>ScaledCurrent = 100ma / $2^{CurrentScale}$<br>Range: -128 .. 127 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-32.2.1 Scaling of attribute values

As part of the Overload object definition, a separate scaling factor is specified current. To maximize the resolution capable or necessary on some devices or applications, current values can be normalized using a binary scale factor before transmission on the bus. Normally, this scaling factor will normally be set up once during initialization depending on the range of current values to be used in the application.

CurrentScale allow the representation of current on the bus to be kept as short as possible while still preserving an acceptable resolution and dynamic range for all applications.

Example:

Output (to device from bus):

TripFLCSet(Overload Object, Attribute ID 3) = 23456

CurrentScale (Overload Object, Attribute ID 12) = 8

$\Rightarrow$ Actual FLC setting used by device = TripFLCSet = $23456 \div 2^8 = 91.625 \times 100mA$ = 9.1625 amps

Input (from drive to bus):

Actual FLC setting used by device = 78.95 amps = 789.5 x 100mA

CurrentScale (Overload Object, Attribute ID 12) = 3

$\Rightarrow$ TripFLCSet(Overload Object, Attribute ID 3) = $789.5 \times 2^8 = 6316$

Devices that do not support CurrentScale (Attributes 12), should return a zero, or not supported attribute error. In this case, the default scaling (0) is presumed. All currents will then default to units of 100ma.

## 5A-32.3 Common Services

**Table 5A-32.3 Overload Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0E hex | Optional | Required | Get_Attributes_Single | Returns the contents of the specified attribute. |
| 10 hex | n/a | Required | Set_Attributes_Single | Modifies an attribute value. |
| 15 hex | n/a | Optional | Restore | Restores attribute values from a storage location accessible by the Save service. This service is only available if the drive is in the Ready_Disabled or Wait_Power states. |
| 16 hex | n/a | Optional | Save | Saves all attribute values to a location accessible by the Restore service. |
| See Appendix A for definition of these services | | | | |

## 5A-32.4 Object-specific Services

The Overload object provides no object specific services.

## 5A-32.5 Behavior

The Overload object behavior can be changed by setting attributes, overload current setting and Trip Class setting will effect the trip curve of the device.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-33 Softstart Object

## Class Code: 2D hex

This object models all the functions specific to a Soft Start Motor Starter.

## 5A-33.1 Class Attributes

**Table 5A-33.1 Softstart Object Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | |

## 5A-33.2 Instance Attributes

**Table 5A-33.2 Softstart Object Instance Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|
| 1 | Optional | Get | NumAttr | USINT | Number of Attributes supported |
| 2 | Optional | Get | Attributes | Array of USINT | List of Attributes supported |
| 3 | Required | Get | AtReference | BOOL | Starting/stopping output voltage reference status<br>0 = Not At Reference<br>1 = Output At Voltage Reference |
| 4 | Required | Set | StartMode | USINT | 0 = No Voltage Ramp<br>No Current Limit<br>1 = Voltage Ramp<br>No Current Limit<br>2 = No Voltage Ramp<br>Current Limit<br>3 = Voltage Ramp<br>Current Limit<br>4 – 9 = Reserved by CIP<br>10 – 255 = Vendor Specific |
| 5 | Optional | Set | StopMode | USINT | 0 = Coast<br>1 = Ramp Down<br>2 = Brake<br>3 – 9 = Reserved by CIP<br>10 – 255 = Vendor Specific |
| 6 | Optional | Get/Set | RampMode | BYTE | 0 = Single Ramp<br>1 = Dual Contiguous Ramp<br>2 = Dual Independent Ramp<br>3 – 9 = Reserved by CIP<br>10 – 255 = Vendor Specific |
| 7 | Optional | Get/Set | RampTime1 | UINT | Tenths of Seconds |
| 8 | Optional | Get/Set | InitialVoltage1 | USINT | 0 - 100 % Of Full Line Voltage |
| 9 | Optional | Get/Set | RampTime2 | UINT | Tenths of Seconds |
| 10 | Optional | Get/Set | InitialVoltage2 | USINT | 0 - 100 % of Full Line Voltage |
| 11 | Optional | Get/Set | Rotation | BOOL | 0 = ABC phase rotation<br>1 = CBA phase rotation |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|
| 12 | Optional | Get/Set | KickStart | BOOL | 0 = disabled<br>1 = enabled fixed time |
| 13 | Optional | Get/Set | KickStartTime | USINT | Tenths of Seconds |
| 14 | Optional | Get/Set | KickStartVoltage | UINT | 0 - 100 % of Full Line Voltage |
| 15 | Optional | Get/Set | EnergySaver | BOOL | 0 = disabled<br>1 = enabled |
| 16 | Optional | Get/Set | DecelTime | UINT | Tenths of Seconds |
| 17 | Optional | Get/Set | CurrentLimitSet | UINT | Percent of Rated Current (Motor Data Instance Attribute 6) |
| 18 | Optional | Get/Set | BrakingCurrentSet | UINT | Percent of Rated Current (Motor Data Instance Attribute 6) |

## 5A-33.3 Common Services

**Table 5A-33.3 Softstart Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0E hex | Optional | Required | Get_Attributes_Single | Returns the contents of the specified attribute. |
| 10 hex | n/a | Required | Set_Attributes_Single | Modifies an attribute value. |
| 15 hex | n/a | Optional | Restore | Restores attribute values from a storage location accessible by the Save service. This service is only available if the drive is in the **Ready_Disabled** or **Wait_Power** states. |
| 16 hex | n/a | Optional | Save | Saves all attribute values to a location accessible by the Restore service. |
| See Appendix A for definition of these services | | | | |

## 5A-33.4 Object-specific Services

The Softstart object provides no object specific services.

## 5A-33.5 Behavior

### 5A-33.5.1 Starting Behavior

The controlled starting of motors can be accomplished using one or more optional approaches; Voltage Ramp (single ramp, dual contiguous ramp, or dual independent ramp options), Current Limit, or a combination of Voltage Ramp /Current Limit.

StartMode (Attribute 4) configures the device to Voltage Ramp / Current Limit starting behavior.

#### 5A-33.5.1.1 Voltage Ramp Starting

Voltage Ramp based starting (with no current limiting) is selected by setting the StartMode (Attribute 4) to a value of “1”. The mode in which the voltage is ramped up is selected by RampMode (Attribute 6).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

When RampMode (Attribute 6) = 0, Single Ramp mode is selected. In this mode the output voltage of the Soft Starter is increased at a fixed rate, over a user defined time period, RampTime1 (Attribute 7), until it reaches the full line voltage rating of the softstart. To over come motor/load inertia, the output voltage may be set to higher than zero voltage, InitialVoltage1 (Attribute 8) before the voltage ramp up is started. For very high load inertia situations, the full line voltage can be applied for a fixed period of time: KickStart (Attribute 13) is set to 1, or optionally a user defined voltage (specified as a % of full line voltage) for a user defined period of time: Kick Start (Attribute 13) is set to 1 , KickStartTime (Attribute 12) (greater than zero), and Kick Start Voltage (Attribute 18) (greater than zero) values are entered.

When RampMode (Attribute 6) = 1, Dual Contiguous Ramp mode is selected. In this mode the output voltage is increased at a fixed rate, over a user defined time period, RampTime1 (Attribute 7), until it reaches InitialVoltage2 (Attribute 10). The output voltage then increases at a different fixed rate, over a second user defined time period, RampTime2 (Attribute 9), until it reaches full line voltage. As in single ramp soft starters, the first ramp of a dual contiguous ramp soft starter may be preceded with Initial Voltage1 and/or Kick Start functions.

When RampMode (Attribute 6) = 2, Dual Independent Ramp mode is selected. In this mode the user can use the Run1 (Control Supervisor Attribute 3) and Run2 (Control Supervisor Attribute 4) commands to alternate between 2 pre downloaded single ramp starting profiles. The use of the Run1 attribute, starts the motor using the RampTime1 and Initial Voltage1 values. The use of the Run2 attribute, starts the motor based on the RampTime2 and InitialVoltage2 values. All other Output Configuration Attributes are common to both ramps, except for potential vendor specific settings of the StopMode (see Stopping Behavior section).

#### 5A-33.5.1.2 Current Limit Starting

Current Limit based starting (with no voltage ramp) is selected by setting the StartMode (Attribute 4) to a value of “2”. In this case the output voltage is increased until the motor current equals the CurrentLimitSet (Attribute 16). The voltage is then maintained at a level that does not allow the current Limit to be exceeded, until either full speed is reached (and motor current drops below the CurrentLimitSet) or until the overload is tripped. The Current Limit may only be exceeded during operation of the Kick Start function. CurrentLimitSet(Attribute 16) is entered as a percent of the Motor Data Object Instance attribute RatedCurrent (Attribute 6) value.

#### 5A-33.5.1.3 Voltage Ramp With Current Limited Starting

Voltage Ramp with Current Limit based starting is selected by setting StartMode (Attribute 4) to a value of “3”. In this case the output voltage increases according to the selected voltage ramp until the CurrentLimitSet (Attribute 16) is reached. At this point, the output voltage stops increasing and current limit based starting continues until either full speed is reached (and motor current drops below the CurrentLimitSet) or until the overload is tripped.

CurrentLimitSet (Attribute 16) is entered as a percent of the Motor Data Object Instance attribute RatedCurrent (Attribute 6) value.

### 5A-33.5.2 Stopping Behavior

Several optional Stop Modes; Coasting, Ramp Down, DC Braking, or Vendor Specific may be selected via StopMode (Attribute 5). The Stop Mode chosen is independent of the devices Start Mode setting.

#### 5A-33.5.2.1 Coasting to Stop

In this mode, the soft starter sets the output to zero volts, and the motor “coasts” to stop.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5A-33.5.2.2 Ramp to Stop

In this mode, the output voltage is decreased at a fixed rate from line voltage to zero volts over the user defined time period Decel Time (Attribute 16).

#### 5A-33.5.2.3 DC Brake to Stop

For fast motor stopping, a DC braking current is applied to the motor for the duration of the DecelTime (Attribute 15). The amount of DC braking current applied to the motor is specified in BrakingCurrentSet (Attribute 17). BrakingCurrentSet (Attribute 17) is entered as a percent of the Motor Data Object Instance attribute RatedCurrent (Attribute 6) value.

#### 5A-33.5.2.4 Vender Specific Stopping

In this mode, the soft starter stops the motor by setting the output voltage according to a profile unique to a particular vendor.

### 5A-33.5.3 Other Attribute Behaviors

#### 5A-33.5.3.1 Rotation (Attribute 11)

Sets the main voltage phase rotation sequence for Phases A, B & C.

#### 5A-33.5.3.2 EnergySaver (Attribute 14)

Sets whether the energy saver function is active when the motor is operating at reference. The energy saving function works as follows: When a motor is operating under a loaded condition and then the load is reduced, the current the motor draws will decrease. The energy saver function lowers the voltage to a point where the current starts to increase again. The output voltage is maintained at that level until the load increases, in which case the soft starter will increase the output voltage back up to the line voltage level.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-33.5.4 Soft Starter Output Voltage Behavior Description

**Figure 5A-33.1 Example: Dual Contiguous Ramp Soft Start With Ramp Down Stop Mode**

**Figure 5A-33.2 Example: Dual Independent Ramp Soft Start With Ramp Down Stop Mode**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-34 Selection Object

## Class Code: 2E hex

The Selection Object manages the selection and distribution of data between objects.

Revision History

**Table 5A-34.1 Selection Object Revision History**

| Revision | Reason for object definition update |
|---|---|
| 01 | Initial Release |
| 02 | Modified for use as general data flow manager and added NV column to attribute table |

## 5A-34.1 Class Attributes

**Table 5A-34.2 Selection Object Class Attributes**

| Number | Need in Implement | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Conditional * | Get | NV | Revision | UINT | Revision of this object | The current value assigned to this attribute is two (2). |
| 2 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | | |
| 8 | Reserved for CIP | | | | | | |
| 9 | Optional | Get | NV | Max number of instances | UINT | Indicates maximum number of instances supported. | Range 0 – 65535. |

*This attribute is REQUIRED if the application supports revision (2) of this object, otherwise the attribute is OPTIONAL.

## 5A-34.2 Class Services

**Table 5A-34.3 Selection Object Class Services**

| Service Code | Need In Implementation | Service Name | Service Description |
|---|---|---|---|
| 08hex | Optional | Create | Used to instantiate a next instance of the class. |
| 09hex | Optional | Delete | Used to delete all instances of class. |
| 06hex | Optional | Start | Used to start all instances of class |
| 07hex | Optional | Stop | Used to stop all instances of class. |
| 05hex | Optional | Reset | Used to reset all instances of class. Clear table entries. |
| 0Ehex | Conditional* | Get_Attribute_Single | Used to read a class attribute value. |

*This Service is REQUIRED if any Class attributes are supported.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-34.3 Instance Attributes

**Table 5A-34.4 Selection Object Instance Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Attribute Name | Data Type | Attribute Description | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Required | Get | V | State | USINT | State of the object. | See attribute description. |
| 2 | Required | Get | NV | Max_ destinations | UINT | Indicates maximum number of destination indexes supported. | Range 0-65535.<br>0 = No destination specified. |
| 3 | Conditional | Get | NV | Number_of_destinations | UINT | Number of entries of destination_list. | Range 0-65535. |
| 4 | Conditional | Set | NV | Destination_list | Array of STRUCT of | List of destination paths | |
| | | | | | {USINT | Pathlength | Path length |
| | | | | | Padded EPATH} | Destination Path | Path. See Appendix C for encoding |
| 5 | Required | Get | NV | Max_ Sources | UINT | Indicates maximum number of source indexes supported. | Range 0-65535.<br>0 = No Source specified. |
| 6 | Required | Get | NV | Number_of_sources | UINT | Number of entries of source_list. | Range 0-65535. |
| 7 | Conditional | Set | NV | Source_list | Array of UINT | Array of consuming I/O Connection Instance IDs. | Connection Instance IDs. |
| 8 | Conditional | Get, Set* | NV | Source_used | UINT | Index into source_list array indicating which source is currently being used. | Range 0-number of sources, where one (1) is first entry in source_list. |
| 9 | Optional | Get | V | Source_ Alarm | Array of BOOL | Indicates sources that are not available. | 0 = Available,<br>1 = Not available,<br>Default values are all 0. |
| 10 | Optional | Get/Set | NV | Algorithm_ Type | USINT | The selection algorithm type. | If not supported, algorithm_type =0. |
| 11 | Conditional | Get/Set | NV | Detection_ Count | USINT | Required when alogrithm type is supported. | Range 2-255. Default value is 4. |
| 12 | Conditional | Get/Set | NV | Selection_ Period | UINT | Required when alogrithm type is supported. | Range 1-65535<br>(ms). |
| 13 | Conditional | Set | NV | object_ source list | ARRAY of STRUCT of | List of source paths | |
| | | | | | { USINT | Pathlength, | Path length |
| | | | | | Padded EPATH } | Source Path | Path. See Appendix C for encoding |
| 14 | Optional | Set | NV | destination_ used | UINT | Index into destination_list array indicating which destination is currently being used | Range 0 to number_of_destinations, where (1) is first entry in destination_list.<br>See Behavior |
| 15 | Conditional | Set* | NV | input_data_ value | determined by source data type | a copy of the currently selected source attribute | See Device Profiles where used for Restrictions on Data Type |
| 16 | Conditional | Get | V | output_data_ value | determined by destination data type | a copy of the currently selected destination attribute | See Device Profiles where used for Restrictions on Data Type |

* An application may restrict access to Get Only, generally based on the Algorithm types supported.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-34.4 Semantics

### 5A-34.4.1 State

Defines the current state of the Selection Object instance. The table below defines the possible states and assigns a value used to indicate that state.

**Table 5A-34.5 Values assigned to the state attribute**

| Value | State Name | Description |
|---|---|---|
| 00 | Non-existent | The Selection Object does not exist. |
| 01 | Idle | The Selection Object does not distribute messages and the internal selection algorithm is not active. |
| 02 | Running | The Selection Object distributes messages and internal selection algorithm is running. |

### 5A-34.4.2 max_destinations

Maximum number of destination paths managed by the Selection Object.

### 5A-34.4.3 number_of_destinations

Number of destination paths managed by the Selection Object. Default value is zero(0). This attribute is Required unless max_destinations = 0.

### 5A-34.4.4 destination_list

This attribute is an array of paths to objects which receive messages from the Selection Object. See Appendix C. All members in the list must be unique. This attribute is Required unless max_destinations = 0.

### 5A-34.4.5 max_sources

Maximum number of  source paths managed by the Selection Object.

### 5A-34.4.6 number_of_sources

Number of source paths managed by the Selection Object. Default value is zero(0). The number_of_sources and number_of_destinations do NOT have to be equal. This attribute is Required unless max_sources = 0.

### 5A-34.4.7 source_list

This attribute is an array of Consuming Connection Instance  IDs which send messages to the Selection Object. All members in the list shall be unique. This attribute is Required unless (a) max_sources = 0, or (b) object_source_list is supported.

### 5A-34.4.8 source_used

This attribute is used to indicate the source of a message which is selected by the Selection Object. The value of this attribute indicates the index into source_list currently in use. A value of zero (0) indicates no source is currently being used.

### 5A-34.4.9 source_alarm

This attribute is used to indicate which sources are available. Size of the array is equal to the value of the max_source attribute.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-34.4.10 algorithm_type

Defines the algorithm types. If this attribute is supported and any value from 01 through 03 is supported, then support of all values, 00 through 03, is required. If values greater than 03 are supported, then support of the values 00 through 03 is optional. If this attribute is not supported, then all messages pass through. See table below.

**Table 5A-34.6 Values assigned to the algorithm_type attribute**

| Value | State Name | Description |
|---|---|---|
| 00 | Pass Through | All messages pass through. |
| 01 | Hot Backup* | Hot Backup is used as a selection algorithm. |
| 02 | First Arrival* | First Arrival is used as a selection algorithm. |
| 03 | Last Arrival* | Last Arrival is used as a selection algorithm. |
| 04 | Programmable Data Flow | Programmable Data Flow is used as a selection algorithm. |
| 05-63 | Reserved | Reserved for future use (Open). |
| 64-C7 | Vendor Specific | |
| C8 -FF | Reserved | Reserved for future use. |
| * If supported, then all values 00 through 03 shall be supported. | | |

### 5A-34.4.11 detection_count

The parameter used by algorithms, which require a counter.

### 5A-34.4.12 selection_period

The parameter used by algorithms, which require a time interval.

### 5A-34.4.13 Object Source List

This attribute is an array of paths from which data is sourced. This attribute is Required unless (a) max_sources = 0, or (b) source_list is supported.

### 5A-34.4.14 Destination Used

This attribute is used to indicate the destination of data. The value of this attribute indicates an index into destination_list currently in use. A value of zero (0) indicates no destination is currently being used.

### 5A-34.4.15 Input Data Value

This attribute contains the value retrieved from the Source. This attribute is Required only when max_sources = 0

### 5A-34.4.16 Output Data Value

This attribute contains the value being delivered to the Destination. This attribute is Required only when max_destinations = 0


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-34.5 Instance Services

**Table 5A-34.7 Selection Object Instance Services**

| Service Code | Need In Implementation | Service Name | Service Description |
|---|---|---|---|
| 0E hex | Required | Get_Attribute_Single | Used to read a Selection Object attribute value. |
| 10 hex | Optional | Set_Attribute_Single | Used to modify a Selection Object attribute value. |
| 06 hex | Optional | Start | Used to start a Selection Object. |
| 07 hex | Optional | Stop | Used to stop a Selection Object. |
| 05 hex | Optional | Reset | Used to reset a Selection Object. Clear paths. |
| 09 hex | Optional | Delete | Used to delete a Selection Object. |
| 18 hex | Optional | Get_Member | Used to get values of a member in an array. |
| 19 hex | Optional | Set_Member | Used to set values to a member in an array. |
| 1A hex | Optional | Insert_Member | Used to insert or add a member to an array. |
| 1B hex | Optional | Remove_Member | Used to remove a member from an array. |

## 5A-34.6 Instance Behavior

The behavior of the Selection Object, while in the RUNNING state, is dependent upon the algorithm_type attribute.

Figure 5A-34.1 The Selection Object State Transition Diagram

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-34.8 State Event Matrix for Selection Object**

| Event | Non-Existent | Idle | Running |
|---|---|---|---|
| Create | Class instantiates a Selection Object. Set default values. Transition to **Idle.** | Not applicable. | Not applicable. |
| Power Up & Configuration Valid | Transition to **Running**. | Not applicable. | Not applicable. |
| Delete | Error: Object does not exist. | Release all associated resources. Transition to **Non-existent.** | Release all associated resources. Transition to **Non-existent.** |
| Start | Error: Object does not exist. | If attributes are valid, Transition to **Running**. | Error: The object cannot perform the requested service in its current mode/state. |
| Stop | Error: Object does not exist. | Error: The object cannot perform the requested service in its current mode/state. | Transition to **Idle**. |
| Reset | Error: Object does not exist. | Error: Set default values. | Discard all received messages. Set default values. Transition to **Idle**. |
| Receive_ Message | Not applicable | Discard the message. | When a message is received from a Connection Object, check source_list attribute and update source_used (if applicable). See algorithm_type for details. If fault detected, set appropriate source index bit in source_alarm attribute. |
| Remove_ Member | Error: Object does not exist. | Remove one member from source_list or destination_list attribute. Subtract one (1) from number_of_sources or number_of_ destinations attribute value. | Remove one member from source_list or destination_list attribute. Subtract one(1) from number_of_sources or number_of_destinations attribute value. |
| Insert_ Member | Error: Object does not exist. | Insert or add one member to source_list or destination_list attribute. Algorithm type shall determine usage of members in array. | Insert or add one member to destination_list attribute. Add one(1) to number_of_ destinations attribute value.<br>A member cannot be added to source_list attribute. In this case, return error. |
| Get_Attribute_Single<br>Set_Attribute_Single<br>Get_Member<br>Set_Member | Error: Object does not exist. | Validate/service the request based on internal logic and per the Access Rules. | Validate/service the request based on internal logic and per the Access Rules. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-34.9 Selection Object Attribute Access**

| Attribute | Non-Existent | Idle | Running |
|---|---|---|---|
| State | Not available | Get only | Get only |
| max_destinations | Not available | Get only | Get only |
| number_of_destinations | Not available | Get only | Get only |
| destination_list | Not available | Get/Set/Insert/Remove | Get/Set/Insert/Remove |
| max_sources | Not available | Get only | Get only |
| number_of_sources | Not available | Get only | Get only |
| source_list | Not available | Get/Set/ Insert /Remove | Get/Set/Remove |
| source_used | Not available | Get only | Get only |
| source_alarm | Not available | Get only | Get only |
| algorithm_type | Not available | Get/Set | Get only |
| detection_count | Not available | Get/Set | Get/Set |
| selection_period | Not available | Get/Set | Get/Set |

### 5A-34.6.1 Message Distribution

The Selection Object manages distribution of messages. This function provides multicast communication among Application Objects. The following diagram illustrates the message flow of this Multicast Communication.

The Selection Object receives a message from a Connection Object and delivers this message to Application Object(s).

**Figure 5A-34.2 I/O Message Distribution**

### 5A-34.6.2 Message Selection

The Selection Object may produce one output message from multiple input messages. This function may be used for redundant node applications. The Selection Object selects one message from more than one message based on its internal algorithm.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

In high reliability systems, there is a requirement for redundant nodes. The Selection Object receives messages from redundant nodes and selects one message based on an internal selection algorithm.

**Figure 5A-34.3 Message Selection Example**

### 5A-34.6.3 Message Selection Algorithms

The following algorithms are currently defined for message selection.

- Pass Through
- Hot Backup
- First Arrival
- Last Arrival
- Programmable Data Flow

#### 5A-34.6.3.1 Pass Through

No algorithm. Forward all arriving messages to destination(s).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-34.6.3.2 Hot Backup

This algorithm is shown in the Figure labeled Hot Backup. Node #1 is higher priority than Node #2. Node #3 normally accepts messages from Node #1. Node failure is detected in the following manner. If the Selection Object detects that Node #1 failed, it accepts messages from Node #2 and forwards them to Application Object(s). Upon the receipt of the message from Node #2, the “Message Counter” within the Selection Object is incremented by one. If the message from Node #1 is received, the “Message Counter” is reset. If the number of the “Message Counter” reaches detection_count (N), the Selection Object detects that Node #1 failed. If Node #1 recovers and resumes sending messages, the Selection Object accepts messages from Node #1. “N” is the configurable maximum count and set in attribute 12. If “number_of_sources”, attribute 7, equals two(2) then the first member in the “source_list”, attribute 8, has the higher priority.

**Figure 5A-34.4 Hot Backup**

### 5A-34.6.3.3 First Arrival

This algorithm is shown in the Figure labeled First Arrival. The Selection Object within Node #3 always accepts the first arriving message from either of the two nodes; Node #1 or Node #2. Upon receipt of the first message from an active node, the “Message Counter” within the Selection Object is incremented by one. The “Message Counter” is decremented each time a message is received from the alternate node. If the number within the “Message Counter” reaches “detection_count”, attribute 12, the Selection Object detects that a node failed. The node to which the positive count is associated determines the node from which the Selection Object forwards the message. “N” is the configurable maximum count contained within attribute 12. The “number_of_sources”, attribute 7, equals two(2). See the Figure labeled Behavior of First Arrival in “Running State” for detail behavior.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5A-34.5 First Arrival**

**Figure 5A-34.6 Behavior of First Arrival in "Running State"**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-34.6.3.4 Last Arrival

This algorithm is shown in the Figure labeled Last Arrival. The Selection Object starts a timer when transmits to running. It forwards the last message received when the timer expires, and immediately restarts the timer. If no message is received AND the timer expires, the Selection Object does not forward a message to the Application Object(s). The “selection_period” which is shown as “dt” in the figure is the value contained in attribute 13. The “number_of_sources”, attribute 7, is equal to two(2) in the example.

**Figure 5A-34.7 Last Arrival**

### 5A-34.6.3.5 Programmable Data Flow

When selected, this algorithm affects the following object behavior:

The Selection object uses the *source-used* attribute to select a Source path from the object_source_list. This specifies the source from which the Selection object instance retrieves data values. Similarly, the *destination_used* attribute is used to select a Destination path from the *destination_list* in order to specify the destination to which the Selection object instance delivers the retrieved data values. Optionally, (as specified by the application) the Selection object may perform some level of data type conversion. If this capability is specified by a Device Profile or Vendor, the extent to which it is supported shall be specified. The combined functions of Retrieval, Data Type Conversion and Delivery constitute the Application Process for the Selection object while executing the Programmable Data Flow algorithm.

A *source_used* value of zero (0) halts the Retrieval portion of the Application Process. The retrieval and subsequent setting of the *input_data_value* is the only behavior effected by halting the Retrieval Application Process (i.e., *source_used* = 0). The *input_data_value* retains the last value set and is then accessible for manual (or I/O) setting. Similarly, a *destination_used* value of zero (0) halts the Delivery portion of the Application Process providing for manual (or I/O) setting of a referenced destination and making available for manual retrieval the *output_data_value*. With both the Retrieval and Delivery Processes halted, the object's Application Process consists only of the (optional) data conversion process, whereby, the *output_data_value* attribute is updated by the object based on the value of the *input_data_value* attribute.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

An attempt to set a member of the *source_list* or *destination_list* to a source or destination EPATH referencing a data type unsupported by the Selection object returns an Invalid Attribute Value error. Similarly, an attempt to set *source_used* and/or *destination_used* to an unsupported value will return an Invalid Attribute Value error; an attempt to set both such that an invalid combination results (i.e., non-matching data types) will return an Object State Conflict.

An attempt to set a member of the *object_source_list* or *destination_list* attributes with an EPATH referencing an object class, instance, attribute or structure element that is not known or is not contained in the processing node returns a Path Destination Unknown error.

**Figure 5A-34.8 Programmable Data Flow Example**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-34.6.4 Examples

**Figure 5A-34.9 I/O Message Distribution**

**Table 5A-34.10 I/O Message Distribution Attribute Values**

| Attribute Number | Attribute Name | Attribute Value |
|---|---|---|
| 1 | state | 02 |
| 3 | number_of_destinations | 3 |
| 4 | destination_list | {Path1, Path2, Path3} |
| 6 | number_of_sources | 01 |
| 7 | source_list | {10} |
| 8 | source_used | 01 |
| 9 | source_alarm | {0} |
| 10 | algorithm_type | 00 |

**Table 5A-34.11 I/O Message Selection Attribute Values**

| Attribute Number | Attribute Name | Attribute Value |
|---|---|---|
| 1 | state | 02 |
| 3 | number_of_destinations | 1 |
| 4 | destination_list | {Path1} |
| 6 | number_of_sources | 02 |
| 7 | source_list | {10,11} |
| 8 | source_used | 01 |
| 9 | source_alarm | {0,0} |
| 10 | algorithm_type | 02 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-35 S-Device Supervisor Object

**Class Code: 30Hex**

This object models the interface, functions and behavior associated with the management of application objects for devices within the "*Hierarchy of Semiconductor Equipment Devices*". Throughout this CIP Standard, objects belonging to this hierarchy are identified as such by a naming convention that includes a prefix of "S-" in the object class name. This "*Hierarchy of Semiconductor Equipment Devices*" is completely defined in this object definition such that all objects belonging to this hierarchy require the existence of an S-Device Supervisor object to manage its functions and behaviors.

The S-Device Supervisor object centralizes application object state definitions and related status information, exception status indications (alarms and warnings), and defines a behavior model which is assumed by objects identified as belonging to the *Hierarchy of Semiconductor Equipment Devices*. That is, if a reset is requested of the S-Device Supervisor object instance, it will be performed by this object instance as well as all of its associated application objects.

Similarly, the Identity object provides an interface to the S-Device Supervisor object. A reset request to the Identity object (of any type) causes a reset request to the S-Device Supervisor object. Further relationships are specified in the Behavior section below.

Additionally, some device attributes are defined which are required in order to specify device models such that they are compliant with the SEMI S/A Network Standard [1], from which the *Hierarchy of Semiconductor Equipment Devices* is derived. Objects defined to exist within the *Hierarchy of Semiconductor Equipment Devices* are done so in order to simplify the management and description of object behavior while insuring compliance with the SEMI Standard.

**NOTE:** By association with this object, the Start, Stop, Reset, Abort, Recover and Perform_Diagnostic Services are inherently supported by all objects within the *Hierarchy of Semiconductor Equipment Devices*. Although, for the associated application object instances, these services are not accessible over the network.

## 5A-35.1 Class Attributes

**Table 5A-35.1 S-Device Supervisor Object Class Attributes**

| Attribute ID | Need in implementation | Access Rule | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|
| 1 | Required | Get | Revision | UINT | = 02 |
| 2 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | |
| 97-98 | Reserved by CIP for Future Use | | | | |
| 99 | Conditional * | Get | Subclass | UINT | Identifies a subset of additional class attributes, services and behaviors. |

* If the value of Subclass is 00 which identifies "no subclass", then this attribute is OPTIONAL in implementation, otherwise, this attribute is REQUIRED.

[1] Semiconductor Equipment and Materials International, Mountain View CA, Standard E54: *Sensor/Actuator Network Common Device Model.*


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-35.1.1 Subclasses

Each class level subclass defines a unique meaning for an overlapping range of class attribute IDs and/or class service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given class is identified by the value of its Subclass class attribute. The subclasses for this object are specified at the end of this object specification section.

## 5A-35.2 Instance Attributes

CIP reserves Attribute ID 100-199 ($64_{hex}$-$C7_{hex}$) for Vendor Defined Attributes. See Chapter 4 for more information on Object Definitions.

**Table 5A-35.2 S-Device Supervisor Object Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Value |
|---|---|---|---|---|---|---|---|
| 1 | Optional | Get | NV | Number of Attributes | USINT | Number of Attributes supported by the object instance | |
| 2 | Optional | Get | NV | Attribute List | Array of USINT | List of attributes supported by the object instance | |
| 3 | Required | Get | NV | Device Type | SHORT STRING | ASCII Text, Max. 8 Characters | See “Semantics” section |
| 4 | Required | Get | NV | SEMI Standard Revision Level | SHORT STRING | Specifies the revision level of the SEMI S/A Network Standard to which the device complies. | For this revision, this attribute must be: “E54-0997” |
| 5 | Required | Get | NV | Manufacturer’s Name | SHORT STRING | ASCII Text, Max. 20 Characters. | See “Semantics” section |
| 6 | Required | Get | NV | Manufacturer’s Model Number | SHORT STRING | ASCII Text, Max. 20 Characters, Manufacturer Specified | |
| 7 | Required | Get | NV | Software Revision Level | SHORT STRING | ASCII Text, Max. 6 Characters | See “Semantics” section |
| 8 | Required | Get | NV | Hardware Revision Level | SHORT STRING | ASCII Text, Max. 6 Characters | See “Semantics” section |
| 9 | Optional | Get | NV | Manufacturer’s Serial Number | SHORT STRING | ASCII Text, Max. 30 Characters, Manufacturer Specified | See “Semantics” section |
| 10 | Optional | Get | NV | Device Configuration | SHORT STRING | ASCII Text, Max. 50 Characters, Manufacturer Specified. Optional additional information about the device configuration. | |
| 11 | Required | Get | V | Device Status | USINT | | See “Semantics” section |
| 12 | Required | Get | V | Exception Status | BYTE | | See “Semantics” section |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## S-Device Supervisor Object, Class Code: $30_{Hex}$

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Value |
|---|---|---|---|---|---|---|---|
| 13 | Conditional based on Exception Status Bit 7 | Get | V | Exception Detail Alarm | STRUCT of: | A Structure of three Structures containing a bit mapped representation of the alarm detail | |
| | | | | Common Exception Detail | STRUCT of: | | |
| | | | | Size | USINT | Number of Common Detail Bytes | |
| | | | | Detail | ARRAY of: | | See “Semantics” section |
| | | | | Detail n | BYTE | | See “Semantics” section |
| | | | | Device Exception Detail | STRUCT of: | | |
| | | | | Size | USINT | Number of Device Detail Bytes | |
| | Conditional based on Size | | | Detail | ARRAY of: | See Device Profile | |
| | Conditional based on Size | | | Detail n | BYTE | See Device Profile | |
| | | | | Manufacturer Exception Detail | STRUCT of: | | |
| | | | | Size | USINT | Number of Manufacturer Detail Bytes | |
| | Conditional based on Size | | | Detail | ARRAY of: | Manufacturer Specified | |
| | Conditional based on Size | | | Detail n | BYTE | Manufacturer Specified | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## S-Device Supervisor Object, Class Code: $30_{Hex}$

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Value |
|---|---|---|---|---|---|---|---|
| 14 | Conditional based on Exception Status Bit 7 | Get | V | Exception Detail Warning | STRUCT of: | A Structure of three Structures containing a bit mapped repre-sentation of the warning detail | |
| | | | | Common Exception Detail | STRUCT of: | | |
| | | | | Size | USINT | Number of Common Detail Bytes | |
| | | | | Detail | ARRAY of: | | See “Semantics” section |
| | | | | Detail n | BYTE | | See “Semantics” section |
| | | | | Device Exception Detail | STRUCT of: | | |
| | | | | Size | USINT | Number of Device Detail Bytes | |
| | Conditional based on Size | | | Detail | ARRAY of: | See Device Profile | |
| | Conditional based on Size | | | Detail n | BYTE | See Device Profile | |
| | | | | Manufacturer Exception Detail | STRUCT of: | | |
| | | | | Size | USINT | Number of Manufacturer Detail Bytes | |
| | Conditional based on Size | | | Detail | ARRAY of: | Manufacturer Specified | |
| | Conditional based on Size | | | Detail n | BYTE | Manufacturer Specified | |
| 15 | Required | Set | NV | Alarm Enable | BOOL | | See “Semantics” section |
| 16 | Required | Set | NV | Warning Enable | BOOL | | See “Semantics” section |
| 17 | Optional | Set | * | Time | DATE_ AND_TIM E | The value of the device’s internal realtime clock. | See “Semantics” section |
| 18 | Optional | Get | NV | Clock Power Cycle Behavior | USINT | Describes the behavior of the device’s internal realtime clock (the Time attribute) during a power cycle | 0 = [default] clock always resets during power cycle<br>1 = clock value is stored in non-volatile memory at power down<br>2 = clock is battery-backed and runs without device power.<br>3-255 = Reserved |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Value |
|---|---|---|---|---|---|---|---|
| 19 | Optional | Get | NV | Last Maintenance Date | DATE | The date on which the device was last serviced. | |
| 20 | Optional | Get | NV | Next Scheduled Maintenance Date | DATE | The date on which it is recommended that the device next be serviced. | |
| 21 | Optional | Get | NV | Scheduled Maintenance Expiration Timer | INT | | See “Semantics” section |
| 22 | Conditional - Required if Calibration Expiration is supported | Set | NV | Scheduled Maintenance Expiration Warning Enable | BOOL | | See “Semantics” section |
| 23 | Optional | Get | NV | Run Hours | UDINT | An indication of the number of hours that the device has had power applied. It has a resolution of 1 hour. This value shall be maintained in nonvolatile memory. | |
| 24 | Optional | Get | V | Endpoint | BOOL | Indicates that the device ha determined that the conditions of its endpoint algorithm are satisfied for a particular recipe step. | 0 = Endpoint not obtained<br>1 = Endpoint obtained<br>See Semantics section |
| 25 | Optional | Set | NV | Recipe | UINT | Provides a common mechanism for selecting endpoint algorithms. | See Semantics section |
| 81-96 | Defined by subclasses below | | | | | | |
| 97-98 | Reserved by CIP for Future Use | | | | | | |
| 99 | Conditional ** | Get | NV | Subclass | UINT | Identifies a subset of additional instance attributes, services and behaviors. | 0 = No subclass<br>1 = Power Generator |

* The nonvolatile behavior of the *Time* attribute is conditional on the value of the *Clock Power Cycle Behavior* attribute.

** If the value of Subclass is 00, which identifies "no subclass" then this attribute is OPTIONAL in implementation, otherwise, this attribute is REQUIRED.

### 5A-35.2.1 Subclasses

Each instance level subclass defines a unique meaning for an overlapping range of instance attribute IDs and/or instance service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given instance is identified by the value of its Subclass instance attribute. The subclasses for this object are specified at the end of this object specification section.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-35.3 Semantics

### 5A-35.3.1 Device Type

The Device Type attribute, identifies the Specific Device Model to which the device is modeled within the *Hierarchy of Semiconductor Equipment Devices*. The value of this string is specified in the SEMI standard suite referenced in the introduction section of this object definition and is represented for reference in the applicable device profile where used.

### 5A-35.3.2 Manufacturer's Name

The Manufacturer's Name attribute, identifies the manufacturer of the device. It is the responsibility of the manufacturer to insure that this ASCII coded text string is sufficiently long to insure uniqueness among manufacturers.

The Device Manufacturer attribute is not guaranteed, by specification, to be unique. Therefore, it is not a substitute for the corresponding attribute of the Identity Object and should not be used for identification purposes.

### 5A-35.3.3 Software Revision Level

This is an ASCII coded text string representing the revision of the software corresponding to the specific device identified by the Identity object and the S-Device Supervisor object.

### 5A-35.3.4 Hardware Revision Level

This is an ASCII coded text string representing the revision of the hardware which is identified by the Identity object and the S-Device Supervisor object. The manufacturer of the device must control this revision such that modifications to the device hardware may be tracked.

### 5A-35.3.5 Manufacturer's Serial Number

This attribute is a string representation of the vendor's serial number of the device, formatted to fit most manufacturing tracking systems. This is not the same as the Identity Object's serial number which is used to uniquely identify the device in the network environment.

### 5A-35.3.6 Device Status

This attribute represents the current state of the device. Its value changes as the state of the device changes. The following values are defined:

**Table 5A-35.3 Device Status Attribute Values**

| Attribute Value | State |
|---|---|
| 0 | Undefined |
| 1 | Self Testing |
| 2 | Idle |
| 3 | Self-Test Exception |
| 4 | Executing |
| 5 | Abort |
| 6 | Critical Fault |
| 7-50 | Reserved by CIP |
| 51-99 | Device Specific |
| 100-255 | Vendor Specific |

### 5A-35.3.7 Exception Status

A single byte attribute whose value indicates that the status of the alarms and warnings for the device. This indication may be provided in one of two methods: Basic or Expanded.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

For the *Basic Method*, bit seven of the Exception Status attribute is set to zero; all exceptions are reported exclusively through communication of this Exception Status attribute. The format of bits zero through six in this mode is device specific; the format may be further specified in an appropriate device profile specification; if it is not specified, then the format of bits zero through six is equivalent to that specified for the expanded method.

For the *Expanded Method*, bit seven of Exception Status attribute is set to one; exceptions are reported through the communication of this Exception Status attribute, formatted as specified in the table below. In addition, the Exception Detail attributes are supported. The Exception Status bits are determined by a logical "OR" of the related Exception Detail bits, as indicated.

**Table 5A-35.4 Exception Status Bit Map**

| | Exception Status Bit Map, Bit 7 set to 0 | Exception Status Bit Map, Bit 7 set to 1 |
|---|---|---|
| **Bit** | **Function** | **Function** |
| 0 | Device Specific definition (See Device Profile) | ALARM/device-common* |
| 1 | | ALARM/device-specific |
| 2 | | ALARM/manufacturer-specific |
| 3 | | reserved -- set to 0 |
| 4 | | WARNING/device-common* |
| 5 | | WARNING/device-specific |
| 6 | | WARNING/manufacturer-specific |
| 7 | 0 == Basic Method | 1 == Expanded Method |

* The alarm or warning is not specific to the device type or device type manufacturer.

### 5A-35.3.8 Exception Detail Alarm and Exception Detail Warning

The formats of these two attributes are identical. Therefore, they are described together here:

Attributes that relate the detailed status of the alarms or warnings associated with the device. Each attribute is a structure containing three members; these three members respectively relate the detailed status of exceptions that are common (i.e., not device-specific), device-specific but not manufacturer-specific, and manufacturer-specific. The common detail is defined below. The device-specific detail is defined in the appropriate Device Profile. The manufacturer-specific detail is defined by the manufacturer. A SIZE value of zero indicates that no detail is defined for the associated exception detail structure.

Each of the three structure members is defined as a structure containing an ordered list (i.e., array) of bytes of length SIZE, and an unsigned integer whose value is SIZE. Each of the bytes in each array has a specific mapping. This mapping is formatted as 8 bits representing 8 independent conditions, whereas a value of 1 indicates that the condition is set (or present), and a value of 0 indicates that the condition is cleared (or not present). Note that if a device does not support an exception detail, the corresponding bit is never set. The bitmaps for alarms and warnings in the corresponding attributes are structured in parallel so that a condition may have either alarm or warning set depending on severity. If a condition inherently cannot be both alarm and warning, then the parallel bit position corresponding to the other state will remain "0."

The existence of an exception detail variable structure is dependent on the value of the Exception Status Attribute; the existence of an exception detail variable structure is only required if bit seven of the Exception Status attribute is set to 1 (indicating Expanded method reporting) and the bit (among bits zero through six) of the Exception Status attribute corresponding to the particular exception type is also set to 1.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-35.3.9 Common Exception Detail

This structure relates exception conditions (i.e., alarms or warnings) which are common to all devices within the Hierarchy of Semiconductor Equipment Devices. The Detail element of the structure is an ordered list (i.e., array) of bytes of length [SIZE] which is the value of the structure element Size. For each byte in the Detail field, all bits which are not identified are reserved for future standardization.

The first byte in this attribute is CommonExceptionDetail[0]. Additional exception details, if provided, are named CommonExceptionDetail[1], . . . CommonExceptionDetail[SIZE]. The specific exception associated with each of the bitmaps is given in the table below. The SIZE for this revision is two (2). The criteria details for each exception condition are outside the scope of this document.

**Table 5A-35.5 Common Exception Detail Attribute Values**

| Bit | Common Exception Detail [0]* | Common Exception Detail [1]* |
|---|---|---|
| 0 | Internal diagnostic exception | power supply overcurrent |
| 1 | Microprocessor exception | reserved power supply |
| 2 | EPROM exception | power supply output voltage |
| 3 | EEPROM exception | power supply input voltage |
| 4 | RAM exception | scheduled maintenance due |
| 5 | Reserved by CIP | notify manufacturer |
| 6 | Internal real-time exception | reset exception |
| 7 | Reserved by CIP | reserved by CIP |

* Note that if a device does not support an exception detail, the corresponding bit is never set

### 5A-35.3.10 Device Exception Detail

This structure, similar in form to Common Exception Detail, relates exception conditions, which are specific to individual devices on the network and are defined in their respective device profiles. The Detail element of the structure is an ordered list (i.e., array) of bytes of length [SIZE], which is the value of the structure element size. For a detailed description of this attribute, consult the appropriate specific device profile.

### 5A-35.3.11 Manufacturer Exception Detail

This structure, similar in form to Common Exception Detail, relates exception conditions, which are specific to the manufacturers of individual devices on the network and are defined by them in their product documentation. The Detail element of the structure is an ordered list (i.e., array) of bytes of length [SIZE], which is the value of the structure element Size. For a detailed description of this attribute, consult the appropriate specific device manufacturer documentation.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-35.6 Exception Detail Format Summary**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Common Exception Detail Size | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Common Exception Detail 0 | Reserved | Real-time Fault | Reserved | Data Memory | Non-Volatile Memory | Code Memory | Micro-processor | Diagnostic |
| Common Exception Detail 1 | Reserved | Reset Exception | Notify Vendor | Scheduled Maint. Due | PS Input Voltage | PS Output Voltage | Reserved | PS Over Current |
| Device Exception Detail Size | x | x | x | x | x | x | x | x |
| Device Exception Detail 0 | — | — | — | — | — | — | — | — |
| Device Exception Detail n | — | — | — | — | — | — | — | — |
| Manufacturer Exception Detail Size | x | x | x | x | x | x | x | x |
| Manufacturer Exception Detail 0 | — | — | — | — | — | — | — | — |
| Manufacturer Exception Detail n | — | — | — | — | — | — | — | — |

### 5A-35.3.12 Alarm Enable and Warning Enable

These Boolean attributes are used to enable (1) or disable (0) the S-Device Supervisor object's process of setting Exception bits. When disabled, corresponding bits are never set; and, if they were set, disabling clears them. Also, alarm and warning states are not retained; when enabled, bits will be set only if the corresponding condition is true.

The default state for these Enable attributes is enabled (1).

### 5A-35.3.13 Time

This optional attribute represents the value of the time and date as maintained by the device's realtime clock with a resolution of one millisecond.

The default value for the Time attribute is zero (0), corresponding to 12:00AM, January 1, 1972, as specified by Appendix C.

### 5A-35.3.14 Scheduled Maintenance Expiration Timer

This attribute, with a resolution of one hour, is used to cause a warning which indicates that a device calibration is due. A S-Device Supervisor timer decrements this attribute once per hour while power is applied. When the attribute is no longer positive and the Scheduled Maintenance Expiration Warning Enable attribute is set to enabled, a Scheduled Maintenance Expiration Warning condition is generated. This causes the Scheduled Maintenance Due Warning bit to be set.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The attribute will not wrap; when the attribute reaches its most negative value, it no longer decrements. The attribute will continue to decrement irrespective of the state of the Scheduled Maintenance Expiration Warning Enable attribute. The value shall be maintained in nonvolatile memory.

### 5A-35.3.15 Scheduled Maintenance Expiration Warning Enable

This Boolean attribute is used to enable (1) or disable (0) the S-Device Supervisor object's process of setting the Scheduled Maintenance Due Exception bit. When disabled, the corresponding bit is never set; and, if it was set, disabling clears it. When enabled, the bit will be set only if the corresponding condition is true.

The default state for this Enable attribute is enabled (1).

### 5A-35.3.16 Endpoint

This attribute is used to indicate that the device has determined that the conditions of its endpoint algorithm are satisfied for a particular recipe step. The vendor must specify the endpoint algorithm, including the configuration mechanisms and parameters.

### 5A-35.3.17 Recipe

This attribute provides a common mechanism for selecting endpoint algorithms. This index pointer is used to specify which endpoint algorithm is currently active. The particular algorithm with its associated configuration, limits and other parameters are manufacturer specific.

## 5A-35.4 Common Services

**Table 5A-35.7 S-Device Supervisor Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0E hex | Conditional | Required | Get_Attributes_Single | Returns the contents of the specified attribute. |
| 10 hex | n/a | Required | Set_Attributes_Single | Modifies an attribute value. |
| 05 hex | n/a | Required | Reset | Resets the device to the Self-Testing state. |
| 06 hex | n/a | Required | Start | Starts the device execution by moving the device to the Executing state. Equivalent to SEMI S/A Network Execute Service |
| 07 hex | n/a | Optional | Stop | Moves the device to the Idle state |
| See Appendix A for definition of these services | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-35.5 Object-specific Services

**Table 5A-35.8 S-Device Supervisor Object Object-specific Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 4B hex | n/a | Required | Abort | Moves the device to the Abort state |
| 4C hex | n/a | Required | Recover | Moves the device out of the Abort state |
| 4E hex | n/a | Required | Perform_Diagnostics | Causes the device to perform a set of diagnostic routines |

**Table 5A-35.9 DS Object Service Parameter Dictionary**

| Parameter | Form | Description of Service |
|---|---|---|
| TestID | USINT | Type and possibly detail of diagnostic test to be performed |

### 5A-35.5.1 Abort

Used to transition the device application objects to the aborted state. This service request may be (and generally will be) originated internally, from application objects.

### 5A-35.5.2 Recover

Used to transition the device application objects, out of the abort state, to the idle state. This service request may be originated internally, from application objects.

### 5A-35.5.3 Perform_Diagnostics

Used to instruct the S-Device Supervisor object to perform a diagnostic test. A diagnostic test is either of type *common* or *device-dependent*. *Common* diagnostic tests could include: RAM, EPROM, non-volatile memory, and communications. The structure of *common* type diagnostic tests are implementation-specific. All detail of *device-dependent* diagnostics is outside the scope of this document.

### 5A-35.5.4 TestID Parameter

The following values are defined for the TestID parameter for the Perform_Diagnostics Service Request:

**Table 5A-35.10 TestID Parameter Attribute Value**

| Attribute Value | State |
|---|---|
| 0 | Standard |
| 1-63 | Reserved |
| 64-127 | Device Specific (defined in Device Profile) |
| 128-255 | Manufacturer Specific (defined by manufacturer) |

Type “Standard” is specified if there is only one type of diagnostic defined or if there are more than one including a type standard. Additional diagnostic types may be defined in the device profile or by the manufacturer.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-35.6 Behavior

### 5A-35.6.1 S-Device Supervisor Object States

**Figure 5A-35.1 DS Object Instance Behavior**

**Table 5A-35.11 DS Instance Behavior State Description**

| State | Description |
|---|---|
| EXECUTING | Device is executing, e.g., it is performing its device specific function. A detailed description of this state is outside the scope of this document. This state may be further specified in an appropriate device profile specification. |
| SELF TESTING | Object instance exists and has been initialized; all attributes have appropriate initial values (as indicated herein and in applicable device profile specification). Exception Status bits have been reset. Device is performing device-specific and device type specific test to determine if it is qualified to execute its application process. |
| SELF TEST EXCEPTION | Object has detected an exception condition during self-testing. The details of the exception are stored in the appropriate attribute values of the S-Device Supervisor object. |
| IDLE | Object and device have been initialized and successfully completed self-testing. Further, the device is not executing the operational components of its device specific functions. |
| ABORT | Object instance is in an aborted state; a detailed description of this state is outside the scope of this specification. |
| CRITICAL FAULT | The object (and device) are in a fault state from which there is no recovery. object services cannot be processed. The conditions required for exit from a critical fault is outside the scope of this specification. |

The S-Device Supervisor Status attribute value indicates the state of the S-Device Supervisor object. It is updated on appropriate state transitions within the S-Device Supervisor object. Attribute values 1 through 6 represent valid states. A value of zero indicates that the S-Device Supervisor state is unknown; conditions under which a zero value may occur are outside the scope of this document.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-35.6.2 State Event Matrix

**Table 5A-35.12 State Event Matrix for S-Device Supervisor Object**

| Event | State | | | | | |
|---|---|---|---|---|---|---|
| | Idle | Self-Testing | Self-Test Exception | Executing | Abort (Recoverable Fault) | Critical Fault |
| Power Applied | — | Default Entry Point: Device performs its Self-Test Application Process | — | — | — | Transition to SELF-TESTING |
| Self-Test Passed | Not Applicable | Transition to IDLE | Not Applicable | Not Applicable | Not Applicable | Not Applicable |
| Self-Test Failed | Not Applicable | Set appropriate Exception Status Bits and Transition to SELF-TEST EXCEPTION | Not Applicable | Not Applicable | Not Applicable | Not Applicable |
| Exception Condition Cleared | Not Applicable | Not Applicable | Set appropriate Exception Status Bits and Transition to SELF-TESTING | Not Applicable | Not Applicable | Not Applicable |
| Critical Fault | Transition to CRITICAL FAULT | Transition to CRITICAL FAULT | Transition to CRITICAL FAULT | Transition to CRITICAL FAULT | Transition to CRITICAL FAULT | Ignore Event |
| Reset Request | Transition to SELF-TESTING | Restart SELF-TESTING | Transition to SELF-TESTING | Transition to SELF-TESTING | Transition to SELF-TESTING | Ignore Event |
| Start Request | Transition to EXECUTING | Error OSC* | Error OSC* | Error AIRS* | Error OSC* | Ignore Event |
| Stop Request | Error AIRS* | Error OSC* | Error OSC* | Transition to IDLE | Error OSC* | Ignore Event |
| Abort Request | Transition to ABORT | Transition to ABORT | Error OSC* | Transition to ABORT | Error AIRS* | Ignore Event |
| Recover Request | Error OSC* | Restart SELF-TESTING | Transition to SELF-TESTING | Error OSC* | Transition to IDLE | Ignore Event |
| Perform Diagnostics Request | Transition to SELF-TESTING | Restart SELF-TESTING | Transition to SELF-TESTING | Transition to SELF-TESTING | Perform all device diagnostics test. | Ignore Event |
| I/O Connection Timeout | Ignore Event | Ignore Event | Ignore Event | Transition to IDLE | Ignore Event | Ignore Event |
| Receipt of First Valid I/O Data | Transition to EXECUTING | Ignore Event | Ignore Event | Normal Response | Ignore Event | Ignore Event |
| I/O Connection Deleted | Ignore Event | Ignore Event | Ignore Event | Transition to IDLE | Ignore Event | Ignore Event |

* Error AIRS = Error Response “Already in Requested Mode/State” (Code 0B hex)
Error OSC = Error Response “Object State Conflict” (Code 0C hex)

Any S-Device Supervisor instance service may be requested internally by the device as specified by the manufacturer. Generally, these requests will be generated as the result of an event such as the activation of a button or external contact closure.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-35.6.3 S-Device Supervisor and Identity Object Interface

The following two tables describe the effects that the Identity object and the S-Device Supervisor object have on each other based on events. This event mapping defines the interface between these two objects.

**Table 5A-35.13 Effect of Identity Object Event**

| Identity Object Event | S-Device Supervisor Object Affected Event |
|---|---|
| Power Applied | Power Applied |
| Failed Tests | Self-Test Failed |
| Passed Tests | Self-Test Passed |
| Major Unrecoverable Fault | Critical Fault |
| Minor Fault | No effect |
| Fault Corrected | Exception Condition Cleared |
| Reset | Reset Request * |
| * The S-Device Supervisor object service requests are generated by the device in response to these events. | |

**Table 5A-35.14 Effect of S-Device Supervisor Object Event**

| S-Device Supervisor Object Event | Identity Object Affected Event |
|---|---|
| Power Applied | Power Applied |
| Self-Test Passed | Passed Tests |
| Self-Test Failed | Failed Tests |
| Exception Condition Cleared | Fault Corrected |
| Critical Fault | Major Unrecoverable Fault |
| Perform Diagnostics Request | No effect. * |
| * The request to perform diagnostics has no effect on the Identity Object. However, the results of the tests may. | |

### 5A-35.6.4 S-Device Supervisor and Application Object Interface

When processing Stop, Start, Reset, Abort, Recover and Perform_Diagnostics service requests, the S-Device Supervisor object coordinates all Application Objects within the device, as appropriate, in order that they will exhibit behavior consistent with the S-Device Supervisor object instance state.

This object interfacing is best described in the SEMI standard suite as referenced in the introduction section of the object definition.

## 5A-35.7 S-Device Supervisor Instance Subclass 01 — Power Generator

The following specification applies to a subclass of this object for application in Power Generator devices such as those used to create plasma energy.

### 5A-35.7.1 Instance Attributes

The following Instance Attributes are specified for this object subclass.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-35.15 Power Generator Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 81 | Conditional* | Set | NV | Energy Control Enabled | BOOL | Controls the Energy Control Application Process (See Behavior) | 0 = Disabled [default]<br>1 = Enabled |
| 82 | Conditional* | Set | NV | Joules to Deliver | UDINT | Controls the Energy Control Application Process (See Behavior) | Joules<br>[default] = 0 |
| 83 | Conditional* | Get | V | Joules Remaining | UDINT | Controls the Energy Control Application Process (See Behavior) | Joules<br>[default] = 0 |
| 84 | Optional** | Set | NV | Active Target ID | UINT | A numerical index referring to which target is being tracked via the Active Target Life Accumulator | [default] = 0 |
| 85 | Optional** | Set | NV | Active Target Life Counter Enable | BOOL | Enables the Active Target Life counter | [default] = 0 |
| 86 | Optional** | Set | NV | Active Target Life | DINT | Once set, the value counts down to zero. Upon reaching a zero count, the associated *Exception Detail Warning* bit is set. | KWH<br>[default] = 0 |
| 87 | Optional | Get | NV | Input Power On Time | UDINT | Factory reset accumulator | Seconds |
| 88 | Optional | Get | NV | Output Power On Time | UDINT | Factory reset accumulator | Seconds |
| 89 | Optional | Set | NV | Filament On Time | UDINT | Settable accumulator | Seconds<br>[default] = 0 |
| 90 | Optional | Get | NV | Total Energy Delivered | UDINT | Factory reset accumulator | KWH |
| 91 | Optional | Get | NV | Power On Cycle Counter | UDINT | Factory reset accumulator | Counts |
| 92 | Optional | Get | NV | Output On Cycle Counter | UDINT | Factory reset accumulator | Counts |
| 93 | Optional | Get | NV | Device Over-temp Event Counter | UDINT | Factory reset accumulator | Counts |
| 94 | Optional | Get | NV | Regulation Alarm Event Counter | UDINT | Factory reset accumulator | Counts |
| 95 | Optional | Set | V | Filament Power Enable | BOOL | Causes the filament power to turn only if the device is in the Executing state | 0 = Disables Power [default]<br>1 = Enables Power |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 96 | Required | Set | V | Output Power Enable | BOOL | Causes the power output to turn on only if the device is in the Executing state | 0 = Disables Power [default]<br>1 = Enables Power |

* Required if Energy Control is supported
** Required if Target Life tracking is supported

### 5A-35.7.2 Services

There are no additional services defined for this subclass

### 5A-35.7.3 Behavior

#### 5A-35.7.3.1 Energy Control Application Process

The Energy Control Application Process makes use of a down counter. When power delivery is started, the Joules Remaining attribute assumes the value of the Joules to Deliver attribute. While power is being delivered, the Energy Accumulator calculates delivered energy and decrements the Joules Remaining attribute accordingly. Upon reaching a value of zero, the power delivery is stopped.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-36 S-Analog Sensor Object

**Class Code: 31 Hex**

The S-Analog Sensor Object models the acquisition of a reading from a physical sensor in a device. Associated with an analog sensor is a reading that has been acquired and corrected with an offset and a gain coefficient, optionally, settable in the object. Additional correction algorithms may be specified by other objects identified in the device profile or as extensions specified by the manufacturer.

This object is a member of the *Hierarchy of Semiconductor Equipment Devices*. As such, its behavior is managed by the S-Device Supervisor Object. See section 0.

## 5A-36.1 Class Attributes

**Table 5A-36.1 S-Analog Sensor Object Class Attributes**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Value |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |
| 32 | Condtional ** | Get | Class Level Status Extension | USINT | Indicates whether a combination or multigauge measures above or below its maximum or minimum measurement range | Bit 0: Reading Invalid<br>Bit 1: Overrange Exceeded<br>Bit 2: Underrange Exceeded |
| 94-96 | Defined by Subclasses | | | | | |
| 97-98 | Reserved by CIP for Future Use | | | | | |
| 99 | Conditional * | Get | Subclass | UINT | Identifies a subset of additional class attributes, services and behaviors. | 0 = No subclass<br>1 = Instance Selector<br>2 - 65535 = Reserved by CIP |

* If the value of Subclass is 00, which identifies "no subclass", then this attribute is OPTIONAL in implementation, otherwise, this attribute is REQUIRED.

** Required for combination gauges and multigauges only

### 5A-36.1.1 Subclasses

Each class level subclass defines a unique meaning for an overlapping range of class attribute IDs and/or class service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given class is identified by the value of its Subclass class attribute. The subclasses for this object are specified at the end of this object specification section.

## 5A-36.2 Instance Attributes

Certain minimal implementations may support any optional “Set” attributes as “Get Only” and still be compliant with this object specification. All required attributes must be supported as specified.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-36.2 S-Analog Sensor Object Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Optional | Get | NV | Number of Attributes | USINT | Number of attributes supported | |
| 2 | Optional | Get | NV | Attribute List | ARRAY OF USINT | List of attributes supported by this object instance | |
| 3 | Optional | See Semantics | NV | Data Type | USINT | Determines the Data Type of *Value* and all related attributes as specified in this table. | see Semantics section |
| 4 | Optional | See Semantics | NV | Data Units | ENGUNITS | Determines the Units context of *Value* and all related attributes. | see Semantics section |
| 5 | Required | Get | V | Reading Valid | BOOL | Indicates that the *Value* attribute contains a valid value. | 0 = invalid<br>1 = valid<br>(invalid: e.g., not warmed up yet) |
| 6 | Required | Get | V | Value | INT or specified by *Data Type* if supported | Analog input value | The corrected, converted, calibrated final value of the sensor.<br>see Semantics section |
| 7 | Required | Get | V | Status | BYTE | Alarm and Warning State of this object instance | see Semantics section |
| 8 | Optional | Set | NV | Alarm Enable | BOOL | Enables the setting of the Alarm Status Bits | 0 = disable [default]<br>1 = enable |
| 9 | Optional | Set | NV | Warning Enable | BOOL | Enables the setting of the Warning Status Bits | 0 = disable [default]<br>1 = enable |
| 10 | Optional | Get | NV | Full Scale | INT or specified by *Data Type* if supported | The *Value* of Full Scale for the sensor. | The value of attribute *Value* corresponding to the Full Scale calibrated measurement of the sensor.<br>[default] = maximum allowable value for the *Data Type* |
| 11 | Optional | Get | NV | Offset-A Data Type | USINT | Determines the Data Type of attribute *Offset-A* | see Semantics section |
| 12 | Optional | Set | NV | Offset-A | INT or specified by *Offset-A Data Type* if supported | An amount added prior to *Gain* to derive *Value* | see Semantics section<br>0 = [default] |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## S-Analog Sensor Object, Class Code: $31_{Hex}$

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 13 | Required if Attribute "Gain" is other than REAL | Get | NV | Gain Data Type | USINT | Determines the Data Type of attribute *Gain* | see Semantics section |
| 14 | Optional | Set | NV | Gain | REAL or specified by *Gain Data Type* if supported | An amount scaled to derive *Value* | see Semantics section<br>1.0 = [default] |
| 15 | Required if Attribute "Gain" is other than REAL | Get | NV | Unity Gain Reference | REAL or specified by *Gain Data Type* if supported | Specifies the value of the *Gain* attribute equivalent to a gain of 1.0 | Used for normalizing the *Gain* attribute.<br>[default] = 1.0<br>e.g., for an UINT type Gain, a Unity Gain Reference may be 10000, allowing a gain of 0.0001 to 6.5535. |
| 16 | Optional | Set | NV | Offset-B | INT or specified by *Data Type* if supported | An amount added to derive *Value* | see Semantics section<br>0 = [default] |
| 17 | Optional | Set | NV | Alarm Trip Point High | INT or specified by *Data Type* if supported | Determines the Value above which an Alarm Condition will occur | see Semantics section<br>[default] = Maximum value for its data type. |
| 18 | Optional | Set | NV | Alarm Trip Point Low | INT or specified by *Data Type* if supported | Determines the Value below which an Alarm Condition will occur | see Semantics section<br>[default] = Minimum value for its data type. |
| 19 | Optional | Set | NV | Alarm Hysteresis | INT or specified by *Data Type* if supported | Determines the amount by which the *Value* must recover to clear an Alarm Condition | see Semantics section<br>[default] = 0 |
| 20 | Optional | Set | NV | Alarm Settling Time | UINT | Determines the time that the *Value* must exceed the Trip Point before the exception condition is generated. | Time in milliseconds<br>see Semantics section<br>[default] = 0 |
| 21 | Optional | Set | NV | Warning Trip Point High | INT or specified by *Data Type* if supported | Determines the *Value* above which a Warning Condition will occur | see Semantics section<br>[default] = Maximum value for its data type. |
| 22 | Optional | Set | NV | Warning Trip Point Low | INT or specified by *Data Type* if supported | Determines the *Value* below which a Warning Condition will occur | see Semantics section<br>[default] = Minimum value for its data type. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## S-Analog Sensor Object, Class Code: $31_{Hex}$

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 23 | Optional | Set | NV | Warning Hysteresis | INT or specified by *Data Type* if supported | Determines the amount by which the *Value* must recover to clear a Warning Condition | see Semantics section<br>[default] = 0 |
| 24 | Optional | Set | NV | Warning Settling Time | UINT | Determines the time that the *Value* must exceed the Trip Point before the exception condition is generated. | Time in milliseconds<br>see Semantics section<br>[default] = 0 |
| 25 | Optional | Set | NV | Safe State | USINT | Specifies the behavior for the *Value* for states other than Execute | see Semantics section<br>[default] = 0 |
| 26 | Optional | Set | NV | Safe Value | INT or specified by *Data Type* if supported | The Value to be used for Safe State = Safe Value | see Semantics section<br>[default] = 0 |
| 27 | Optional | Set | NV | Autozero Enable | BOOL | Enables the Autozero | see Semantics section<br>0 = disable [default]<br>1 = enable |
| 28 | Optional | Get | V | Autozero Status | BOOL | Indicates the status of the automatic nulling | see Semantics section<br>[default] = 0 |
| 29 | Optional | Set | NV | Autorange Enable | BOOL | Enables the automatic range switching | see Semantics section<br>0 = disable [default]<br>1 = enable |
| 30 | Optional | Get | V | Range Multiplier | REAL | Indicates the current range multiplier | see Semantics section<br>[default] = 1.0 |
| 31 | Optional | Set | NV | Averaging Time | UINT | Specifies the time over which analog samples are averaged. | Time in Milliseconds of a moving-window average.<br>0 = disable averaging [default]<br>Values less than the sample rate of the device also disable averaging. |
| 32 | Optional | Get | NV | Overrange | INT or specified by *Data Type* if supported | Specifies the highest valid *Value* | The value above which attribute *Reading Valid* is set to invalid.<br>[default] = maximum allowable value for the *Data Type* |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 33 | Optional | Get | NV | Underrange | INT or specified by *Data Type* if supported | Specifies the lowest valid *Value* | The value below which attribute *Reading Valid* is set to invalid.<br>[default] = minimum allowable value for the *Data Type* |
| 34 | Optional | Set | NV | Produce Trigger Delta | INT or specified by *Data Type* if supported and/or *Produce Trigger Delta Type* | The amount by which *Value* must change before a Change of State Production is triggered | 0 = Disabled [default]<br>See Semantics section |
| 35 | Conditional * | Set | NV | Calibration Object Instance | UINT | Indicates which Calibration object instance is active for this object | 0 = Disabled [default]<br>See Semantics section |
| 36 | Optional | Set | NV | Produce Trigger Delta Type | USINT | Specifies the Type for Produce Trigger Delta | 0 = [default]<br>See Semantics section |
| 37 | Optional | Set | NV | Value Descriptor | SHORT STRING | A user defined ASCII value used to identify the sensed parameter | Maximum of 20 characters |
| 79-96 | Defined by Subclasses below | | | | | | |
| 97-98 | Reserved by CIP for Future Use | | | | | | |
| 99 | Conditional ** | Get | NV | Subclass | UINT | Identifies a subset of additional instance attributes, services and behaviors. | 0 = No subclass<br>1 = Flow Diagnostics<br>2 = Heat Transfer Vacuum Gauge<br>3 = Capacitance Manometer<br>4 = Cold Cathode Ion Gauge<br>5 = Hot Cathode Ion Gauge<br>6 = Transfer Function<br>7-65535=Reserved by CIP |

* See Semantics section.

** If the value of Subclass is 00, then this attribute is OPTIONAL in implementation, otherwise, this attribute is REQUIRED.

### 5A-36.2.1 Subclasses

Each instance level subclass defines a unique meaning for an overlapping range of instance attribute IDs and/or instance service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given instance is identified by the value of its Subclass instance attribute. The subclasses for this object are specified at the end of this object specification section.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-36.3 Semantics

### 5A-36.3.1 Data Type

All Data Type attributes, including *Data Type*, *Offset-A Data Type* and *Gain Data Type*, use the enumerated values specified in Appendix C-6.1.

The *Data Type* attribute is settable only in the *Idle State* and only if no attribute belonging to the object instance is the endpoint of an I/O connection in the *Established State*.

The *Data Type* attribute may change automatically based upon established I/O connections. See Behavior section for more information on this mechanism.

### 5A-36.3.2 Data Units

Specifies the context of *Value* and related attributes (such as, offset and trip points) for this object instance. See Appendix D for a list of values. A request to set attribute to an unsupported value will return an error response.

The *Data Units* attribute is settable only in the *Idle State*.

### 5A-36.3.3 Value, Offset (A and B) and Gain

An S-Analog Sensor object instance derives a reading from a physical analog sensor. The reading is converted to the data type and units specified for the ***Value*** attribute. The ***Offset-A, Offset-B*** and ***Gain*** attributes are applied to the sensor reading as specified by the following formula:

*Value* = *Gain* • ( Sensor Reading + *Offset-A* ) + *Offset-B*

Typically, the *Offset-A* or *Offset*-B attributes are modified by the Zero-Adjust service and the *Gain* attribute is modified by the Gain_Adjust services; particularly, when the device utilizes a non-linear conversion algorithm. However, support of these services is not required. See Behavior section.

### 5A-36.3.4 Status

A bit mapped byte which indicates the Alarm and Warning Exception status of the object instance. The following definition applies:

**Table 5A-36.3 Status Attribute Bit Definitions**

| Bit | Definition |
|---|---|
| 0 | High Alarm Exception: 0 = cleared; 1 = set |
| 1 | Low Alarm Exception: 0 = cleared; 1 = set |
| 2 | High Warning Exception: 0 = cleared; 1 = set |
| 3 | Low Warning Exception: 0 = cleared; 1 = set |
| 4 | Reserved |
| 5 | Reserved |
| 6 | Reserved |
| 7 | Reserved |

### 5A-36.3.5 Trip Points, Hysteresis and Settling Time

Trip Point High is the level above which the *Value* attribute will cause an Alarm or Warning exception condition.

Trip Point Low is the level below which the *Value* attribute will cause an Alarm or Warning exception condition.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

A Hysteresis value specifies the amount by which the Value attribute must transition in order to clear an Alarm or Warning condition. For example: A Trip Point High value of 100 and a hysteresis value of 2 will result in an exception condition being set when the Value is above 100 and cleared when the Value drops below 98. Similarly, A Trip Point Low value of 100 and a hysteresis value of 2 will result in an exception condition being set when the Value is below 100 and cleared when the Value increases above 102.

The Settling Time determines the amount of time that the Value attribute must exceed the Trip Point before the exception condition is generated. The Settling Time also applies to the clearing of the condition.

### 5A-36.3.6 Safe State

This attribute specifies what value will be held in *Value* for states other than Executing. See the S-Device Supervisor object definition in Section 0 for a description of object states. The purpose of this mechanism is to allow other devices, that may be using this *Value*, to transition to, or remain in, a safe state in the event of this device transitioning to a FAULT, IDLE, or ABORT state. The following values are defined:

**Table 5A-36.4 Safe State Attribute Values**

| Attribute Value | State |
|---|---|
| 0 | Zero |
| 1 | Full Scale |
| 2 | Hold Last Value |
| 3 | Use Safe Value |
| 4 | Continue Sensing |
| 5-50 | Reserved |
| 51-99 | Device Specific |
| 100-255 | Vendor Specific |

### 5A-36.3.7 Safe Value

For Safe State set to Use Safe Value, this attribute holds the value to which the *Value* attribute will be set for object instance states other than Executing.

### 5A-36.3.8 Autozero Enable and Autozero Status

When the autozero is enabled, the device will automatically invoke a Zero_Adjust service request (no parameter) contingent upon a set of conditions specified by the manufacturer. These conditions may be determined by the value of an attribute (e.g., setpoint) or some other mechanism defined by the manufacturer. See Zero_Adjust service.

While the device is in the process of nulling due to an Autozero event, the value of *Autozero Status* is one (1). When the device is not in the process of nulling, this value is zero (0).

### 5A-36.3.9 Autorange Enable and Range Multiplier

When the autorange is enabled, the device will automatically switch full-scale range based on a set of conditions specified by the manufacturer. The Range Multiplier indicates the range scale.

An example of how Autorange may work is: when the *Value* is less than 9% with a *Range Multiplier* of 1.0, the *Range Multiplier* switches to 10.0 (the *Value* then reads 90% of the 10X range). When the *Value* then reaches 100% with a *Range Multiplier* of 10.0, the *Range Multiplier* returns to 1.0 (the *Value* then reads 10% of the 1X range).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-36.3.10 Produce Trigger Delta

This attribute is used in conjunction with the "Change of State" production trigger type. Upon transition of the associated connection object instance (any Change of State connection pointing to the S-Analog Sensor object *Value* attribute) to the established state, a production is immediately triggered and this reported *Value* is stored internally for the determination of the next production trigger. When the *Value* changes by an amount of at least the *Produce Trigger Delta* (i.e., the *Value* as compared to the internally stored previously produced Value), a new production is triggered, and this reported *Value* becomes the new internally stored Value for the determination of the next production trigger. The interpretation of this attribute is absolute value unless otherwise specified by the *Produce Trigger Delta Type* attribute.

### 5A-36.3.11 Calibration Object Instance

This attribute is used to select an instance of the S-Gas Calibration object or the S-Sensor Calibration object as specified by the device profile where this object is used. The selected S-Calibration object instance provides the data with which an S-Analog Sensor object instance enacts the appropriate calibration algorithm for a given gas type, material type or other parameter dependency that might affect a sensor's calibration.

A Set_Attribute_Single request, specifying a value not supported, will return an "invalid attribute value" error response. A list of acceptable values for this attribute is derived from a class level service request to the Calibration object.

* Conditionally Required: If a device profile specifies a Calibration object relationship for an S-Analog Sensor object instance, then this attribute is required. See the S-Gas Calibration object or the S-Sensor Calibration object definition for more information.

### 5A-36.3.12 Produce Trigger Delta Type

The *Produce Trigger Delta Type* attribute specifies the interpretation of the *Produce Trigger Delta* attribute. The following values are defined:

**Table 5A-36.5 Produced Trigger Delta Type Attribute Values**

| Produce Trigger Delta Type Attribute Value | Produce Trigger Delta Type | Produce Trigger Delta Data Type |
|---|---|---|
| 0 [default] | Absolute Value | As Specified by Data Type |
| 1 | Percent (see below) | UINT |
| 2 - 255 | Reserved by CIP for Future Use | |

Due to the logarithmic, or other non-linear nature of some analog measurements, the data units of the *Produce Trigger Delta* attribute may be set to a Percent *Produce Trigger Delta*. This also causes the Data Type of *Produce Trigger Delta* to be type UINT with a resolution of 0.1%.

For example, if the last COS-produced reading was 5E-5, the *Produce Delta Trigger Type* was set to 1 (Percent), and the *Produce Trigger Delta* = 200 (which equates to a delta of 20%), then the next reading will be produced at 4E-5 or 6E-5.

### 5A-36.3.13 Value Descriptor

The Value Descriptor attribute identifies the Value of the S-Analog Sensor Object. It describes the physical meaning of the value the sensor delivers, e.g.: ion current, electron density or intensity of one wavelength of an optical emission spectrum.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-36.4 Common Services

The S-Analog Sensor Object provides the following Common Services:

**Table 5A-36.6 S-Analog Sensor Object Common Services**

| Service | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| Code | Class | Instance | | |
| 0E hex | Conditional * | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 01 hex | n/a | Required | Set_Attribute_Single | Modifies an attribute value. |
| *The Get_Attribute_Single service is REQUIRED if any attributes are implemented. | | | | |
| See Appendix A for definition of these services | | | | |

## 5A-36.5 Object–specific Services

The S-Analog Sensor Object provides the following Object-Specific Services:

**Table 5A-36.7 S-Analog Sensor Object Object-specific Services**

| Service | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| Code | Class | Instance | | |
| 4B hex | n/a | Optional | Zero_Adjust | Causes the device to modify attribute *Offset-A* and/or *Offset-B* such that attribute *Value* equals the Target Value sent with the request. |
| 4C hex | n/a | Optional | Gain_Adjust | Causes the device to modify attribute *Gain*, such that attribute *Value*, equals the Target Value sent with the request. |

### 5A-36.5.1 Zero_Adjust and Gain_Adjust Services

The Zero_Adjust and Gain_Adjust services are used to cause the S-Analog Sensor Object device to modify its *Offset-A* and/or *Offset-B* and *Gain* attribute values based upon manufacturer specific algorithms. The target value specified in the service request represents the actual parametric measurement that the physical sensor should be reporting at the time of the request.

There are no state transitions associated with the invocation of these services. It is, therefore, incumbent upon the user to establish the device into the desired configuration prior to, and during, the execution of these services. This will generally involve exposing the sensor to a known environment and treating the values read during execution of the services accordingly.

A success service response indicates that the service was accepted and the application process started.

**Table 5A-36.8 Zero_Adjust Request Service Data Field Parameters**

| Parameter | Required | Data Type | Description | Semantics of Values |
|---|---|---|---|---|
| Target Value | Optional | Specified by the value of attribute *Data Type* | The target value for the zero calibration | The value to which the *Value* attribute will be set. If not specified, the default value of zero is used. |

**Table 5A-36.9 Gain_Adjust Request Service Data Field Parameters**

| Parameter | Required | Data Type | Description | Semantics of Values |
|---|---|---|---|---|
| Target Value | Required | Specified by the value of attribute *Data Type* | The target value for the gain calibration | The value to which the *Value* attribute will be set. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-36.6 Behavior

The behavior of this object is managed by the S-Device Supervisor Object in Section 5-35.5.

An S-Analog Sensor object instance acquires a reading from a physical sensor, as identified by the application of the object, and applies an algorithm to modify the reading into the appropriate *Date Type* and *Data Units*. Optionally, additional corrective algorithms are applied to further correct for various calibration effects. These additional algorithms are specified in other objects, as identified in the device profile, or as extensions, specified by the manufacturer.

All Full Scale, Trip Point, Overrange and Underrange calculations, as specified above, utilize the *Value* attribute.

### 5A-36.6.1 Data Type

If the implementation of this object specifies more than one valid Data Type value, in the device profile or by vendor, then the following behavior with respect to *Data Type* applies: The Data Type value will be set automatically based upon the first valid I/O connection established by the device. If, however, a device is specified by a vendor to support only one Data Type, this behavior is not supported.

If no established I/O connections exist, which include an attribute from this object, then the *Data Type* attribute is settable provided that the object is in the *Idle State*. The following example demonstrates this behavior:

A device profile specifies an instance of the S-Analog Sensor object as well as two static Assembly object instances, both with data attribute components mapped to this object instance. Assembly object instance ID 1 specifies INT data types and Assembly object instance ID 2 specifies REAL data types.

After the device is On-Line, it is configured with an I/O connection to Assembly instance ID 2. When the connection transitions to the *Established State*, this object instance attribute *Data Type* is automatically set with the value for REAL before any data is communicated to, or from, the object instance. Any subsequent attempt to connect to Assembly instance ID 1 would then be rejected and result in an INVALID ATTRIBUTE VALUE error with the additional error code indicating the ID of the offending attribute, which in this case would be the connection path.

## 5A-36.7 S-Analog Sensor Object Class Subclass 01 (Instance Selector)

The following is a **class-level subclass** extension to the S-Analog Sensor Object. It provides a single port mechanism for flowing data from the active S-Analog Sensor Instance and allowing that the Active Instance ID may change. The intended application is for Gauge Controllers where more than one gauge is connected, but only one gauge is considered the "active" gauge.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-36.7.1 Class Attributes

The following Class Attributes are specified for this object subclass.

**Table 5A-36.10 Instance Selector Subclass Class Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 94 | Optional | Get | V | Active Value | INT or specified by Data Type (Instance Attribute ID 3) if supported | Can be used by assemblies to produce this class-level attribute, instead of the Value (Attribute ID 6) of the S-Analog Sensor Instances. | |
| 95 | Optional | Get | V | Active Instance Number | UINT | Identifies the object instance that is providing the *Value* which is copied into *the Active Value* for all input Assemblies and the Alarm/Warning Exception Details for the S-Device Supervisor object.<br>See Behavior section. | Default = 1 |
| 96 | Optional | Get | NV | Number of Gauges | USINT | Identifies the number of gauge instances present in the device. | default = [gauge-specific] |

### 5A-36.7.2 Semantics

#### 5A-36.7.2.1 Active Value

Assemblies or connections may produce this class-level attribute, instead of the *Value* (Attribute ID 6) of the active S-Analog Sensor instance. The S-Analog Sensor class-level attribute *Active Instance Number* identifies the object instance that is currently active and providing the *Value* to the *Active Value* class-level attribute which is, in turn, produced by the input assemblies that have *Active Value* as a member.

#### 5A-36.7.2.2 Active Instance Number

The device internally modifies this attribute, as required, to identify the S-Analog Sensor object instance providing the *Value* member which is copied into the *Active Value* for all Input Assemblies and the Alarm/Warning Exception Details for the S-Device Supervisor object.

See Behavior for more information on the mechanism.

#### 5A-36.7.2.3 Number of Gauges

This attribute is used to determine the size of all Input Assemblies within a node. See the respective Device Profile for its usage within a Device Type.

### 5A-36.7.3 Services

There are no additions or restrictions to the Object Services for this object subclass.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-36.7.4 Behavior

The specific algorithm used to set the *Active Instance Number* is manufacturer specific. However, most will follow the basic logic that each of multiple gauges are valid (or ideally suited) for specific ranges of measurement. Therefore, the *Active Instance Number* will be modified based upon the *Active Value* in order that the best gauge, corresponding to a given S-Analog Sensor instance, will be active for the given measurement range.

## 5A-36.8 S-Analog Sensor Object Instance Subclass 01 (Flow Diagnostics)

The following specification applies to a subclass of this object for application in Mass Flow Controller devices.

### 5A-36.8.1 Instance Attributes

The following Instance Attributes are specified for this object subclass.

**Table 5A-36.11 Flow Diagnostics Subclass Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 95 | Optional | Set | NV | Flow Totalizer | ULINT | Total gas flowed through the device since this value was last set to zero | Units are Standard CCs<br>see Behavior section<br>default = 0 |
| 96 | Optional | Set | NV | Flow Hours | UDINT | Total time device has been powered and flowing gas since this value was last set to zero | Resolution is one hour<br>see Behavior section<br>default = 0 |

### 5A-36.8.2 Services

There are no additions or restrictions to the Object Services for this object subclass.

### 5A-36.8.3 Behavior

#### 5A-36.8.3.1 Flow Totalizer and Flow Hours Process

The factory configured out-of-box values for the Flow Totalizer and Flow Hours attributes are both zero. The attributes are only modifiable with *set_attribute_single* service requests; they are not altered by the *Reset* service, including power-cycle, of either the Identity or the S-Device Supervisor objects.

The Flow Totalizer attribute is incremented, at a rate of once every cubic centimeter of gas flow, by the S-Analog Sensor object instance to reflect the amount of gas that has flowed through the device. Upon reaching its maximum value, the Flow Totalizer value is no longer incremented and remains at its maximum value.

The Flow Hours attribute is incremented, at a rate of once every hour, by the S-Analog Sensor object instance to reflect the amount of time that gas has flowed through the device. This condition is determined by the *Value* attribute being greater than 0.5% of full scale. Upon reaching its maximum value, the Flow Hours value is no longer incremented and remains at its maximum value.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-36.9 S-Analog Sensor Object Instance Subclass 02 (Heat Transfer Vacuum Gauge)

Following is a subclass extension to the S-Analog Sensor Object. The original usage for the subclass is for Heat Transfer Vacuum Gauges. The Heat Transfer Vacuum Gauge extension is used in conjunction with the Vacuum/Pressure Gauge Profile providing control and status information unique to heat transfer gauges - Convection, Pirani and Thermocouple gauge types.

### 5A-36.9.1 Instance Attributes

Certain minimal implementations may support any optional ”Set” attributes as ”Get” only and still be compliant with this object specification. Although no attribute is individually required, this object requires that one or more attributes be used.

**Table 5A-36.12 Heat Transfer Vacuum Gauge Subclass Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 91 | Optional | Set | NV | Cable Length | UINT | Transducer cable length for compensation, if required | [default = 0] Units in cm see Semantics |
| 92 | Optional | Set | NV | Sensor High Temp Warning Value | REAL | Maximum compensatable Sensor Temperature. | [default = 0] Units in °C |
| 93 | Optional | Get | V | Sensor Temperature | REAL | Sensor temperature in °C. | see Semantics |
| 94 | Optional | Get | V | Sensor Warning | STRUCT of BYTE BYTE | Bit definitions of Sensor Warning | [default=0] see Semantics |
| 95 | Optional | Get | V | Sensor Alarm | STRUCT of BYTE BYTE | Bit definitions of Sensor Alarms | [default=0] see Semantics |
| 96 | Optional | Get | V | Status Extension | BYTE | Bit-mapped byte providing additional status bits of an S-Analog Sensor instance. | [default=0] see Semantics |

### 5A-36.9.2 Semantics

#### 5A-36.9.2.1 Cable Length

This attribute specifies the cable length of the transducer cable (cable between the sensor [filament] and its electronic).

#### 5A-36.9.2.2 Sensor Temperature

Ambient temperature of the sensor housing. This value could be used for temperature compensation.

#### 5A-36.9.2.3 Sensor Warning

The Sensor Warning attribute provides 16 bits for the current Sensor Warning state of the object. The Sensor Warning attribute also maps to the S-Device Supervisor Exception Detail Warning attribute. The following table provides definitions for each Sensor Warning bit. Reserved bits must be set to zero.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-36.13 Sensor Warning Attribute**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Sensor Warning Byte 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| Sensor Warning Byte 1 | High Warning Exception | Low Warning Exception | Reserved | Reserved | Reserved | Sensor High Temperature Warning | Electronics Warning | Reserved |

### 5A-36.9.2.4 Sensor Alarm

The Sensor Alarm attribute provides 16 bits for the current Sensor Alarm state of the object. The Sensor Alarm attribute also maps to the S-Device Supervisor Exception Detail Alarm attribute. The following table provides definitions for each Sensor Alarm bit. Reserved bits must be set to zero.

**Table 5A-36.14 Sensor Alarm Attribute**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Sensor Alarm Byte 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Sensor Element Failure |
| Sensor Alarm Byte 1 | High Alarm Exception | Low Alarm Exception | Reserved | Reserved | Reserved | Over Temperature of Electronics | Electronics Failure | Reserved |

### 5A-36.9.2.5 Status Extension

8 Bits providing the current sensor alarm state of the object.

**Table 5A-36.15 Status Extension Attribute**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Reserved | Reserved | Reserved | Reserved | Reserved | Underrange Exceeded | Overrange Exceeded | Reading Invalid* |
| * Note: Logical inversion of Reading Valid | | | | | | | |

### 5A-36.9.3 Services

There are no additions or restrictions to the Object Services for this object subclass.

### 5A-36.9.4 Behavior

There are no additions or restrictions to the Object Behavior for this object subclass.

## 5A-36.10 S-Analog Sensor Object Instance Subclass 03 (Diaphragm Gauge)

Following is a subclass extension to the S-Analog Sensor Object. The original usage for the subclass is for Diaphragm Gauge Gauges. The Diaphragm Gauge extension is used in conjunction with the Vacuum/Pressure Gauge Profile providing control and status information.

### 5A-36.10.1 Instance Attributes

Certain minimal implementations may support any optional ”Set” attributes as ”Get” only and still be compliant with this object specification. Although no attribute is individually required, this object requires that one or more attributes be used.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-36.16 Diaphragm Gauge Subclass Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 92 | Optional | Set | V | Sensor Temperature Actual | REAL | The temperature reading of the sensor in degrees Celsius | degrees Celsius |
| 93 | Optional | Set | NV | Sensor Temperature Setpoint | REAL | The temperature in Celsius that the sensor must reach to have a valid reading | degrees Celsius [default] = vendor specific See 5A-36.10.4 for behavior |
| 94 | Optional | Get | V | Sensor Warning | STRUCT of BYTE BYTE | Bit definitions of Sensor Warnings | [default=0] See Semantics |
| 95 | Optional | Get | V | Sensor Alarm | STRUCT of BYTE BYTE | Bit definitions of Sensor Alarms | [default=0] See Semantics |
| 96 | Optional | Get | V | Status Extension | BYTE | Bit mapped byte providing additional status of an S-Analog Sensor instance. | [default=0] See Semantics |

## 5A-36.10.2 Semantics

### 5A-36.10.2.1 Sensor Warning

The Sensor Warning attribute provides 16 bits for the current Sensor Warning state of the object. The Sensor Warning attribute also maps to the S-Device Supervisor Exception Detail Warning attribute as specified in the Vacuum/Pressure Gauge device profile. The following table provides definitions for each Sensor Warning bit. Reserved bits must be set to zero.

**Table 5A-36.17 Sensor Warning Attribute**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Sensor Warning Byte 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Not At Temperature |
| Sensor Warning Byte 1 | High Warning Exception | Low Warning Exception | Reserved | Reserved | Reserved | Reserved | Electronics Warning | Reserved |

### 5A-36.10.2.2 Sensor Alarm

The Sensor Alarm attribute provides 16 bits for the current Sensor Alarm state of the object. The Sensor Alarm attribute also maps to the S-Device Supervisor Exception Detail Alarm attribute as specified in the Vacuum/Pressure Gauge device profile. The following table provides definitions for each Sensor Alarm bit. Reserved bits must be set to zero.

**Table 5A-36.18 Sensor Alarm Attribute**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Sensor Alarm Byte 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Diaphragm Failure |
| Sensor Alarm Byte 1 | High Alarm Exception | Low Alarm Exception | Reserved | Reserved | Reserved | Over Temperature of Electronics | Electronics Failure | Reserved |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5A-36.10.2.3 Status Extension

8 Bits providing the current sensor alarm state of the object.

**Table 5A-36.19 Status Extension Attribute**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Reserved | Reserved | Reserved | Reserved | Reserved | Underrange Exceeded | Overrange Exceeded | Reading Invalid* |

* Note: Logical inversion of Reading Valid

### 5A-36.10.3 Services

There are no additions or restrictions to the Object Services for this object subclass.

### 5A-36.10.4 Behavior

For heated sensors, the ***Sensor Temperature*** attribute indicates the temperature at which the sensor has warmed up. Once the sensor has ”warmed up”, the *Reading Valid* attribute of the S-Analog Sensor Object will be allowed to be set to TRUE provided that all other conditions governing its behavior are also met. The warning bit in the Sensor Warning attribute for *Not At Temperature* shall be cleared.

## 5A-36.11 S-Analog Sensor Object Instance Subclass 04 (Cold Cathode Ion Gauge)

Following is a subclass extension to the S-Analog Sensor Object. The original usage for the subclass is for Cold Cathode Ion Vacuum Gauges. The Cold Cathode Ion Vacuum Gauge extension is used in conjunction with the Vacuum/Pressure Gauge Profile providing control and status information.

A Cold Cathode Ion Vacuum Gauge is a device that measures pressure at vacuum levels. At low pressures the operating of these gauges depends upon the establishment of a gas discharge between two metal electrodes by the application of a sufficiently high d.c. voltage. The gas discharge current is pressure dependent.

### 5A-36.11.1 Instance Attributes

Certain minimal implementations may support any optional ”Set” attributes as ”Get” only and still be compliant with this object specification.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-36.20 Cold Cathode Ion Guage Subclass Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 91 | Optional | Set | V | High Voltage | REAL | High voltage applied to the anode. | Volts<br>[default] = vendor specific |
| 92 | Optional | Set | NV | Sensitivity | REAL | Conversion Factor from current ratio to pressure | [default] = 1.0<br>see Semantics section |
| 93 | Required | Get | V | High Voltage Status | BOOL | Indicates whether the high voltage is turned on or off. | 0 = Off [default]<br>1 = On |
| 94 | Optional | Get | V | Sensor Warning | STRUCT of<br>BYTE<br>BYTE | Bit definitions of Sensor Warnings | [default=0]<br>see Semantics |
| 95 | Optional | Get | V | Sensor Alarm | STRUCT of<br>BYTE<br>BYTE | Bit definitions of Sensor Alarms | [default=0]<br>see Semantics |
| 96 | Optional | Get | V | Status Extension | BYTE | Bit mapped byte providing additional status of an S-Analog Sensor instance | [default=0]<br>see Semantics |

## 5A-36.11.2 Semantics:

### 5A-36.11.2.1 High Voltage

Valid values for the High Voltage attribute are manufacturer specific. Attempts to set the High Voltage to an invalid value shall return the Invalid Attribute Value error response.

### 5A-36.11.2.2 Sensitivity

The conversion factor used to convert the ratio of gauge currents to pressure. This conversion is manufacturer specific.

### 5A-36.11.2.3 Sensor Warning

The Sensor Warning attribute provides 16 bits for the current Sensor Warning state of the object. The Sensor Warning attribute also maps to the S-Device Supervisor Exception Detail Warning attribute as specified in the Vacuum/Pressure Gauge device profile. The following table provides definitions for each Sensor Warning bit. Reserved bits must be set to zero.

**Table 5A-36.21 Sensor Warning Attribute**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Sensor Warning Byte 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| Sensor Warning Byte 1 | High Warning Exception | Low Warning Exception | Reserved | Reserved | Overpressure High Voltage Off | No Ignition Detected | Electronics Warning | Reserved |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-36.11.2.4 Sensor Alarm

The Sensor Alarm attribute provides 16 bits for the current Sensor Alarm state of the object. The Sensor Alarm attribute also maps to the S-Device Supervisor Exception Detail Alarm attribute as specified in the Vacuum/Pressure Gauge device profile. The following table provides definitions for each Sensor Alarm bit. Reserved bits must be set to zero.

**Table 5A-36.22 Sensor Alarm Attribute**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Sensor Alarm Byte 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| Sensor Alarm Byte 1 | High Alarm Exception | Low Alarm Exception | Reserved | Reserved | Overpressure High Voltage Off | Over Temperature of Electronics | Electronics Failure | Reserved |

### 5A-36.11.2.5 Status Extension

8 Bits providing the current sensor alarm state of the object.

**Table 5A-36.23 Status Extension Attribute**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Reserved | Reserved | Reserved | Reserved | Reserved | Underrange Exceeded | Overrange Exceeded | Reading Invalid* |

* Note: Logical inversion of Reading Valid

## 5A-36.11.3 Services

The following instance-level services are defined for this subclass of the S-Analog Sensor object:

**Table 5A-36.24 Cold Cathode Ion Gauge Subclass Instance Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| $62_{hex}$ | n/a | Conditional * | Set High Voltage State | Sets the high voltage to the state specified by the High Voltage State parameter. |
| $63_{hex}$ | n/a | Conditional * | Clear Overpressure High Voltage Off Alarm | Clears the Overpressure High Voltage Off bit after an automatic high voltage off |

* Required for single gauges only, otherwise support is optional

### 5A-36.11.3.1 Clear Overpressure High Voltage Off Alarm Service

If the high voltage is switched off automatically in case of an overpressure condition, it is not possible to switch on the high voltage again before the Overpressure High Voltage Off bit is reset. This service attempts to reset the Overpressure High Voltage Off bit. See the State Event Matrix below.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-36.11.3.2 Set High Voltage State Service

Attempts to set the high voltage to the state specified by the HighVoltageState parameter. This service may fail depending upon the current object state. See the State Event Matrix below.

**Table 5A-36.25 Set High Voltage State Service Data Field Parameters**

| Parameter | Required | Data Type | Description | Semantics of Values |
|---|---|---|---|---|
| HighVoltageState | Required | BOOL | See Behavior | 0 = switches high voltage OFF<br>1 = switches high voltage ON |

## 5A-36.11.4 Behavior

### 5A-36.11.4.1 Alarms/Warnings

If any bit of the attribute ”Sensor Alarm” is set, the high voltage is switched off automatically.

### 5A-36.11.4.2 No ignition detection

The “No Ignition Detected” bit in the Sensor Warning attribute will be set under the following conditions:

The discharge is undetectable during normal operation.

Ignition is not detected following an Explicit request to set the High Voltage ON attribute to On.

These conditions are likely to arise when the gauge is operating at very low pressures.

**Figure 5A-36.1 S-Analog Sensor Object Instance Behavior**

See the S-Device Supervisor object in Section 0 for more information on the Executing State.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-36.26 S-Analog Sensor Sub-State Event Matrix**

| EVENT | SUB – STATE | | |
|---|---|---|---|
| | **High Voltage OFF** | **High Voltage ON** | **Sensor Alarm High Voltage Off** |
| Set High Voltage State ON Request | Transition to *High Voltage ON* | Error AIRS * | Error OSC ** |
| Set High Voltage State OFF Request | Error AIRS * | Transition to *High Voltage OFF* | Error OSC ** |
| Electronics Failure | Transition to *Sensor Alarm High Voltage Off* | Transition to *Sensor Alarm High Voltage Off* | Event reported in Sensor Alarm |
| Overtemperature | Transition to *Sensor Alarm High Voltage Off* | Transition to *Sensor Alarm High Voltage Off* | Event reported in Sensor Alarm |
| Clear Overpressure High Voltage Off Alarm Request (Condition: *ONLY* Overpressure High Voltage Off in Sensor Alarm set) | Not Applicable | Not Applicable | Transition to *High Voltage OFF* |
| Clear Overpressure High Voltage Off Alarm Request (Condition: Overpressure Emission Off and other Sensor Alarms set) | Not Applicable | Not Applicable | Error OSC ** |
| Clear Overpressure High Voltage Off Alarm Request (Sensor Alarms not set) | Error AIRS * | Error OSC ** | Not Applicable |

* Error AIRS = Error Response "Already in Requested Mode/State" (Code $0B_{hex}$)
** Error OSC = Error Response "Object State Conflict" (Code $0C_{hex}$)

## 5A-36.12 S-Analog Sensor Object Instance Subclass 05 (Hot Cathode Ion Gauge)

Following is a subclass extension to the S-Analog Sensor Object. The original usage for the subclass is for Hot Cathode Ion Vacuum Gauges. The Hot Cathode Ion Vacuum Gauge extension is used in conjunction with the Vacuum/Pressure Gauge Profile providing control and status information.

A Hot Cathode Ion Vacuum Gauge is a device that measures pressure at vacuum levels. It contains a hot filament for the generation of free electrons which are accelerated (emission current) and, by collision, generate ionized gas molecules. The positive ionized molecules are attracted to, and collected by, a negatively charged collector (collector current). The ratio of the two currents (collector to emission) is proportional to the pressure.

Generally, these devices support a "degas" mode of operation by a resistive method or an electron bombardment method.

### 5A-36.12.1 Instance Attributes

Certain minimal implementations may support any optional "Set" attributes as "Get" only and still be compliant with this subclass specification. All required attributes must be supported as specified.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-36.27 Hot Cathode Ion Gauge Subclass Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 79 | Optional | Get | V | Filament Bias Voltage | REAL | Diagnostic readout of filament bias voltage | Volts<br>[default] = vendor specific |
| 80 | Optional | Set | V | Grid Voltage | REAL | Setting of grid voltage. | Volts<br>[default] = vendor specific |
| 81 | Conditional<br>See Semantics | Set | V | Active Degas Filament | BYTE | Indicates which degas filament is selected. | Bit mapped byte of up to 8 filaments as specified by the Vendor.<br>See Semantics |
| 82 | Optional | Set | NV | Degas Power | REAL | Indicates degas power in watts | Power in Watts<br>[default] = vendor specific |
| 83 | Optional | Get | V | Filament Current | REAL | Indicates actual filament current in A | Filament Current in amps.<br>[default] = vendor specific |
| 84 | Optional | Get | V | Degas Time Off Remaining | UINT | Remaining time, in seconds, for the current degas off cycle. | [default] = 0<br>See Semantics |
| 85 | Optional | Set | NV | Degas Time Off Interval | UINT | Minimum time, in seconds, between consecutive degas functions. | [default] = vendor specific<br>See Semantics |
| 86 | Optional | Get | V | Degas Time On Remaining | UINT | Remaining time, in seconds, for the current degas on cycle. | [default] = 0<br>See Semantics |
| 87 | Optional | Set | V | Degas Time On Interval | UINT | Indicates the length of time, in seconds, that degas is on. | [default] = vendor specific<br>See Semantics |
| 88 | Optional | Get | V | Degas Status | BOOL | Indicates current degas state. | 0 = Off<br>1 = On |
| 89 | Conditional<br>See Semantics | Set | NV | Active Filament | BYTE | Indicates which filament(s) is/are selected. | Bit mapped byte of up to 8 filaments as specified by the Vendor.<br>See Semantics |
| 90 | Optional | Set | NV | Sensitivity | REAL | Conversion Factor from current ratio to pressure | [default] = 1.0<br>see Semantics section |
| 91 | Optional | Set | NV | Emission Current | REAL | Indicates setting level of emission current in amps. Can be indicated in either continuous or discrete readings allowed by the vendor. | Vendor specifies the range of supported values and the [default]<br>Note: Degas Mode may override this setting. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 92 | Optional | Set | NV | Mode Filament Selection | BOOL | Selects automatic or user filament selection | see Semantics<br>0 = automatic<br>1 = user [default] |
| 93 | Required | Get | V | Emission Status | BOOL | Indicates whether the emission is turned on or off. | 0 = Off<br>1 = On |
| 94 | Optional | Get | V | Sensor Warning | STRUCT of<br>BYTE<br>BYTE | Bit definitions of Sensor Warnings | [default=0]<br>see Semantics |
| 95 | Optional | Get | V | Sensor Alarm | STRUCT of<br>BYTE<br>BYTE | Bit definitions of Sensor Alarms | [default=0]<br>see Semantics |
| 96 | Optional | Get | V | Status Extension | BYTE | Bit-mapped byte providing additional status bits of an S-Analog Sensor instance. | [default=0]<br>see Semantics |

## 5A-36.12.2 Semantics

### 5A-36.12.2.1 Active Degas Filament

Indicates which degas filament is active. 0 = off / 1 = on. Filament selection may be made either automatically by the device, or remotely by setting this attribute.

**Table 5A-36.28 Active Degas Filament Attribute**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Degas Filament 8 | Degas Filament 7 | Degas Filament 6 | Degas Filament 5 | Degas Filament 4 | Degas Filament 3 | Degas Filament 2 | Degas Filament 1 |

### 5A-36.12.2.2 Degas Time Off Interval and Degas Time Off Remaining

The Degas Time Off Interval attribute specifies the minimum time, in seconds, between degas cycles. The valid range for this attribute is manufacturer specific.

The Degas Time Off Remaining attribute provides the remaining time, in seconds, before a new degas cycle can be started. When the Degas Status attribute transitions to the Off state, the Degas Time Off Remaining attribute is set to the value of the Degas Time Off Interval attribute. The device then decrements the value of the Degas Time Off Remaining attribute at one second intervals until the attribute value reaches zero. The “Object State Conflict” error response will be returned for all Set Degas State ON requests received while the value of the Degas Time Off Remaining attribute is greater than zero.

### 5A-36.12.2.3 Degas Time On Interval and Degas Time On Remaining

The *Degas Time On Interval* attribute specifies the maximum time, in seconds, for degas. The manufacturer may limit the maximum value for this attribute.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The *Degas Time On Remaining* attribute provides the remaining time, in seconds, for the current degas cycle. When the *Degas Status* attribute transitions to the On state, the *Degas Time On Remaining* attribute is set to the value of the *Degas Time On Interval* attribute. The device decrements the value of the *Degas Time On Remaining* attribute at one second intervals and stops the active degas cycle when the attribute value reaches zero. Whenever the *Degas Status* attribute transitions to the Off state from the On state, e.g. alarm detected, Set Degas State request received, etc., the *Degas Time On Remaining* attribute is set to the value zero.

### 5A-36.12.2.4 Active Filament

Indicates which degas filament is active. 0 = off / 1 = on. Filament selection may be made either automatically by the device, or remotely by setting this attribute.

**Table 5A-36.29 Active Filament Attribute**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Sensor Filament 8 | Sensor Filament 7 | Sensor Filament 6 | Sensor Filament 5 | Sensor Filament 4 | Sensor Filament 3 | Sensor Filament 2 | Sensor Filament 1 |

### 5A-36.12.2.5 Sensitivity

The conversion factor used to convert the ratio of gauge currents to pressure using the following formula:

$$P = (1/S)\ (C/E)$$

Where:
P = pressure (*Value* attribute)
S = sensitivity (*Sensitivity* attribute)
C = collector current (measured by the device, Amps)
E = emission current (*Emission Current* attribute, Amps)

Sensitivity units are in inverse pressure units.

### 5A-36.12.2.6 Mode Filament Selection

The Mode Filament Selection attribute determines whether devices with multiple filaments automatically select the active filament(s) for Emission and Degas ON, or whether the user manually selects the filament(s).

#### 5A-36.12.2.6.1 Automatic Filament Selection

With “Automatic Filament Selection” (*Mode Filament Selection* attribute = 0), the device determines, using a manufacturer specific algorithm, which filament to use for Emission ON and Degas ON. When the device is configured for automatic filament selection, the *Active Filament* and *Active Degas Filament* attributes have Get access.

If a filament(s) is broken, the corresponding bit(s) in the *Sensor Warning* attribute is set, and the device selection algorithm must avoid selecting the broken filament(s).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5A-36.12.2.6.2 User Filament Selection

With "User Filament Selection" (*Mode Filament Selection* attribute = 1), the user selects which filament to use for Emission ON and Degas ON. When the device is configured for user filament selection:

1. The Active Filament attribute has Set access.
2. The Active Degas Filament attribute has Set access if the device supports individual selection of the Emission and Degas ON filaments. If the device uses the same filament for Emission and Degas ON, the Active Degas Filament has Get access.
3. If the user attempts to select a Filament that is broken, the device returns an "Invalid Attribute Value" error response.

For devices that support User Filament Selection Mode and which provide multiple filaments, the *Active Filament* and *Active Degas Filament* (if the device supports degas) attributes are required. The behavior of a device with more than one filament on is device specific. Generally, this will not be permitted and attempts to do so will return an error.

### 5A-36.12.2.7 Sensor Warning

The Sensor Warning attribute provides 16 bits for the current Sensor Warning state of the object. The Sensor Warning attribute also maps to the S-Device Supervisor Exception Detail Warning attribute as specified in the Vacuum/Pressure Gauge device profile. The following table provides definitions for each Sensor Warning bit. Reserved bits must be set to zero.

**Table 5A-36.30 Sensor Warning Attribute**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Sensor Warning Byte 0 | Sensor Filament 8 Warning | Sensor Filament 7 Warning | Sensor Filament 6 Warning | Sensor Filament 5 Warning | Sensor Filament 4 Warning | Sensor Filament 3 Warning | Sensor Filament 2 Warning | Sensor Filament 1 Warning |
| Sensor Warning Byte 1 | High Warning Exception | Low Warning Exception | Reserved | Sensor Warning * | Pressure Too High For Degas | Reserved | Electronics Warning | Exceeded Maximum C/E Ratio |

* Bit 4 of Byte 1, "Sensor Warning" relates to error conditions of the filament (cathode), anode, and ion collector.

### 5A-36.12.2.8 Sensor Alarm

The Sensor Alarm attribute provides 16 bits for the current Sensor Alarm state of the object. The Sensor Alarm attribute also maps to the S-Device Supervisor Exception Detail Alarm attribute as specified in the Vacuum/Pressure Gauge device profile. The following table provides definitions for each Sensor Alarm bit. Reserved bits must be set to zero.

**Table 5A-36.31 Sensor Alarm Attribute**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Sensor Alarm Byte 0 | Sensor Filament 8 Alarm | Sensor Filament 7 Alarm | Sensor Filament 6 Alarm | Sensor Filament 5 Alarm | Sensor Filament 4 Alarm | Sensor Filament 3 Alarm | Sensor Filament 2 Alarm | Sensor Filament 1 Alarm |
| Sensor Alarm Byte 1 | High Alarm Exception | Low Alarm Exception | Reserved | Environment Failure * | Overpressure Emission Off | Over Temperature of Electronics | Electronics Failure | Exceeded Maximum C/E Ratio |

* Bit 4 of Byte 1, "Environment Failure" relates to environment error conditions of the filament (cathode), anode, and ion collector which could cause an automatic emission off.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-36.12.2.9 Status Extension

8 Bits providing the current sensor alarm state of the object.

**Table 5A-36.32 Status Extension Attribute**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Reserved | Reserved | Reserved | Reserved | Reserved | Underrange Exceeded | Overrange Exceeded | Reading Invalid* |

* Note: Logical inversion of Reading Valid

## 5A-36.12.3 Services

The following instance-level services are defined for this subclass of the S-Analog Sensor object:

**Table 5A-36.33 Hot Cathode Ion Gauge Subclass Instance Attributes**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| $61_{hex}$ | n/a | Optional | Set Degas State | Activate/deactivates degas mode according to the parameter Degas State. Degas mode may be terminated either automatically by device timeout, or remotely by this service. |
| $62_{hex}$ | n/a | Conditional * | Set Emission State | Turns the filament on and off according to the parameter Emission State |
| $63_{hex}$ | n/a | Conditional * | Clear Emission Off Alarm | Clears the Alarm attributes after an automatic ”emission off”. This service acknowledges and attempts to reset the Overpressure Emission Off and Exceeded Maximum C/E Ratio and Environment Failure Alarm attributes before an *Emission State ON* can be applied. |

* Required for single gauges only, otherwise support is optional

### 5A-36.12.3.1 Clear Emission Off Alarm Service

If the emission is switched off automatically in case of an overpressure or any other environmental (for example glow discharge) or Exceeded Maximum C/E Ratio condition, it is not possible to switch on the emission again before the related Alarm attribute is reset. This service resets the Alarm attributes: Overpressure Emission Off and Exceeded Maximum C/E Ratio and Environment Failure. See the State Event Matrix in section 5A-36.11.2 for additional information.

### 5A-36.12.3.2 Set Emission State

**Table 5A-36.34 Set Emission State Service Data Field Parameters**

| Parameter | Required | Data Type | Description | Semantics of Values |
|---|---|---|---|---|
| EmissionState | Required | BOOL | See Behavior | 0 = switches Emission OFF<br>1 = switches Emission ON |

### 5A-36.12.3.3 Set Degas State

**Table 5A-36.35 Set Degas State Service Data Field Parameters**

| Parameter | Required | Data Type | Description | Semantics of Values |
|---|---|---|---|---|
| DegasState | Required | BOOL | See Behavior | 0 = switches Degas OFF<br>1 = switches Degas ON |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-36.12.4 Behavior

#### 5A-36.12.4.1 Sensor Alarms/Warnings: Filaments

When the *Mode Filament Selection* attribute is set to "User or Automatic Filament Selection ", if any bit of the *Sensor Alarm* attribute is set, the emission is switched off automatically. If a filament is broken, the corresponding bit in the *Sensor Warning* attribute is set.

When the *Mode Filament Selection* attribute is set to "***User Filament Selection***", if the selected filament is broken, the corresponding bits in both the *Sensor Warning* and *Sensor Alarm* attributes are set and the device transitions to the state *Sensor Alarm Emission Off* . The alarm is cleared by selecting a new active filament and the device transitions to the state *Emission Off*. In this case the warning bit remains set and the alarm bit gets cleared.

When the *Mode Filament Selection* attribute is set to "***Automatic Filament Selection***", the device sets all warning bits of all sensor filaments that are broken or failed. If the selected filament is broken, the device selects a new filament automatically and sets the warning bit of the new failed filament. If all filaments are broken, the corresponding bits in the *Sensor Alarm* attribute are set and the device transitions to the state Sensor Alarm Emission Off

#### 5A-36.12.4.2 Sensor Alarm Byte 1:

##### 5A-36.12.4.2.1 Bit 0: Exceeded Maximum C/E Ratio

In case of an Exceeded Maximum C/E Ratio the device transitions to the state Sensor Alarm Emission Off. This error condition has to be acknowledged before the emission could be switched on again. This acknowledge is done by the service *Clear Emission Off Alarm*. After receiving this service, the device transitions to the state *Emission Off*.

##### 5A-36.12.4.2.2 Bit 1: Electronics Failure

The device has detected an internal electronics failure which could not be cleared. The emission is switched off and the device transitions to the state *Sensor Alarm Emission Off*.

##### 5A-36.12.4.2.3 Bit 2: Over Temperature of Electronics

The devices temperature is above its specified range. The emission is switched off and the device transitions to the state *Sensor Alarm Emission Off* and remains there until the temperature is below the critical limit. If the user attempts to switch on the emission while this bit is set, the error response Object State Conflict is returned. If the temperature goes below the critical limit, the device automatically transitions to the state *Emission Off*.

##### 5A-36.12.4.2.4 Bit 3: Overpressure Emission Off

In case of an overpressure emission off, the user has to acknowledge the reason of the Emission Off situation. The acknowledge is done by the service: *Clear Emission Off Alarm*. After receiving this service, the device transitions to the state Emission Off.

##### 5A-36.12.4.2.5 Bit 4: Environment Failure

Some other reasons than an overpressure condition (for example grid error or emission current being lost to a plasma) may result in an automatic emission switch off. This bit is set after any kind of these clearable emission switch off conditions. In this case the device transitions to the state *Sensor Alarm Emission Off*. The service *Clear Emission Off Alarm* is used to acknowledge the error condition and the device transitions to the state Emission Off.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-36.12.4.3 Sensor Warning Byte 1:

#### 5A-36.12.4.3.1 Bit 3: Pressure Too High For Degas

If the pressure is above the manufacturer-specified range at which degas is allowed, this bit will be set. This bit is cleared if the pressure is below the manufacturer specified range at which degas is allowed.

### 5A-36.12.5 S-Analog Sensor Object Sub-State for the Executing State

**Figure 5A-36.2 S-Analog Sensor Object Instance Behavior**

See the S-Device Supervisor object (Section 0) for more information on the Executing State.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-36.12.6 S-Analog Sensor Sub-State Event Matrix

**Table 5A-36.36 S-Analog Sensor Sub-State Event Matrix**

| Event | Sub-State | | | | |
|---|---|---|---|---|---|
| | **Emission OFF** | **Emission ON No Degas Range** | **Emission ON Degas Range** | **Sensor Alarm Emission Off** | **Degas On** |
| Electronics Failure | Transition to *Sensor Alarm Emission Off* | Transition to *Sensor Alarm Emission Off* | Transition to *Sensor Alarm Emission Off* | Event reported in Sensor Alarm | Transition to *Sensor Alarm Emission Off* |
| Overtemperature | Transition to *Sensor Alarm Emission Off* | Transition to *Sensor Alarm Emission Off* | Transition to *Sensor Alarm Emission Off* | Event reported in Sensor Alarm | Transition to *Sensor Alarm Emission Off* |
| Temperature falls below overtemperature limit | Not Applicable | Not Applicable | Not Applicable | Transition to *Emission Off* | Not Applicable |
| Environmental Failure like glow discharge | Not Applicable | Transition to *Sensor Alarm Emission Off* | Transition to *Sensor Alarm Emission Off* | Not Applicable | Transition to *Sensor Alarm Emission Off* |
| Exceeded Maximum C/E Ratio | Not Applicable | Transition to *Sensor Alarm Emission Off* | Transition to *Sensor Alarm Emission Off* | Not Applicable | Transition to *Sensor Alarm Emission Off* |
| Clear Emission Off Alarm Request (Condition: **ONLY** Overpressure Emission Off and/or Sensor Failure and/or Exceeded Maximum C/E Ratio in Sensor Alarm set) | Not Applicable | Not Applicable | Not Applicable | Transition to *Emission OFF* | Not Applicable |
| Clear Emission Off Alarm Request (Condition: Overpressure Emission Off and/or Sensor Failure and/or Exceeded Maximum C/E Ratio **and** other Sensor Alarms set) | Not Applicable | Not Applicable | Not Applicable | *Error OSC* ** | Not Applicable |
| Filament Alarm | Not Applicable | Transition to *Sensor Alarm Emission Off* | Transition to *Sensor Alarm Emission Off* | Not Applicable | Transition to *Sensor Alarm Emission Off* |
| Clear Emission Off Alarm Request (Sensor Alarms not set) | *Error AIRS* * | *Error OSC* ** | *Error OSC* ** | Not Applicable | *Error OSC* ** |
| Set Emission State On Request and current pressure <= minimum degas pressure | Transition to *Emission ON Degas Range* | *Error AIRS* * | *Error AIRS* * | *Error OSC* ** | *Error AIRS* * |
| Set Emission State On Request and current pressure > minimum degas pressure | Transition to *Emission ON No Degas Range* | *Error AIRS* * | *Error AIRS* * | *Error OSC* ** | *Error AIRS* * |
| Set Emission State On Request at a broken filament in case of user filament selection | *Error OSC* ** | *Error AIRS* * | *Error AIRS* * | *Error OSC* ** | *Error OSC* ** |
| Emission State On Request all filaments broken | Not Applicable | Not Applicable | Not Applicable | *Error OSC* ** | Not Applicable |
| Current Pressure <= minimum degas pressure | Ignore Event | Transition to *Emission ON Degas Range* | Ignore Event | Ignore Event | Ignore Event |
| Current Pressure <= minimum degas pressure | Ignore Event | Transition to *Emission ON Degas Range* | Ignore Event | Ignore Event | *Ignore Event* |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Event | Sub-State | | | | |
|---|---|---|---|---|---|
| | Emission OFF | Emission ON No Degas Range | Emission ON Degas Range | Sensor Alarm Emission Off | Degas On |
| Current Pressure > minimum degas pressure | Ignore Event | Ignore Event | Transition to *Emission ON No Degas Range* | Ignore Event | Transition to *Emission ON No Degas Range* |
| Valid Set Degas State On Request | Error OSC ** | Error OSC ** | Transition to *Degas On* | Error OSC ** | Error AIRS * |
| Set Degas State On Request at a broken filament in case of user filament selection | Error OSC ** | Error OSC ** | Error OSC ** | Error OSC ** | Error AIRS * |
| Set Degas State On Request and *Degas Time Off Remaining* attribute > 0 | Error OSC ** | Error OSC ** | Error OSC ** | Error OSC ** | Not Applicable |
| Set Degas State Off Request | Error OSC ** | Error OSC ** | Error OSC ** | Error OSC ** | Transition to *Emission ON Degas Range* |
| *Degas Time On Remaining* attribute decrements to zero | Not Applicable | Not Applicable | Not Applicable | Not Applicable | Transition to *Emission ON Degas Range* |

* Error AIRS = Error Response "Already in Requested Mode/State" (Code $0B_{hex}$)
** Error OSC = Error Response "Object State Conflict" (Code $0C_{hex}$)

## 5A-36.13 S-Analog Sensor Object Instance Subclass 06 - Transfer Function

The following specification applies to a subclass of this object.

### 5A-36.13.1 Instance Attributes

**Table 5A-36.37 Transfer Function Subclass Instance Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 96 | Optional | Set | NV | Pressure Variable Mapping Function | USINT | Select function modifier of variable source to convert to selected units | 0 = None (linear)<br>1 = Log()<br>2 = Exp()<br>3 = Interpolate, manufacturer specific.<br>4-127 = Reserved by CIP<br>128-255 = Vendor Specific<br>See Semantics section for usage<br>[default] = 0 |

### 5A-36.13.2 Semantics

#### 5A-36.13.2.1 Pressure Variable Mapping Function

This attribute specifies an added function to the analog sensor object for the pressure variable sensor. This is to allow flexibility in connecting non-intelligent pressure devices to the Process Control Valve. The object uses:

Value = Gain (Sensor Reading + Offset-A ) + Offset-B

Which is now embellished as:

Value = Gain (Function(Sensor Reading + Offset-A) ) + Offset-B

Where Function() is one of the following for each attribute value:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

0: Value = Gain (Sensor Reading + Offset-A) + Offset-B

1: Value = Gain (Log10(Sensor Reading + Offset-A) ) + Offset-B

2: Value = Gain (Exp10(Sensor Reading + Offset-A) ) + Offset-B

3: Value = Gain (Interp(Sensor Reading + Offset-A) ) + Offset-B


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-37 Analog Actuator Object

**Class Code: 32 Hex**

The S-Analog Actuator Object models the interface to a physical actuator in a device. Associated with an analog actuator is a value which is corrected with an offset and a gain coefficient, optionally settable in the object before it is output to the physical actuator. Manufacturers may specify additional correction algorithms as extensions to this object.

The context of this object is that of an "Actuator" which should not be confused with a "Valve". In particular, ZERO corresponds to the default powered-off state or typically NOT ACTUATED. For example, an S-Analog Actuator object physically connected to a *Normally Open* Valve, would result in the *Valve* being *ON* (i.e.: open or flowing) when the *Actuator* is *ZERO* (i.e.: not actuated).

Additionally, the S-Analog Actuator Object provides two sets of trip-point definitions. The behavior associated with these trip points is described in sections below.

This object is a member of the *Hierarchy of Semiconductor Equipment Devices*. As such, its behavior is managed by the S-Device Supervisor Object. See Section 0.

## 5A-37.1 Class Attributes

**Table 5A-37.1 S-Analog Actuator Object Class Attributes**

| Attribute ID | Need in implementation | Access Rule | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | |
| 97-98 | Reserved by CIP for Future Use | | | | |
| 99 | Conditional * | Get | Subclass | UINT | Identifies a subset of additional class attributes, services and behaviors. |

* If the value of Subclass is 00, which identifies "no subclass", then this attribute is OPTIONAL in implementation, otherwise, this attribute is REQUIRED.

### 5A-37.1.1 Subclasses

Each class level subclass defines a unique meaning for an overlapping range of class attribute IDs and/or class service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given class is identified by the value of its Subclass class attribute. The subclasses for this object are specified at the end of this object specification section.

## 5A-37.2 Instance Attributes

Certain minimal implementations may support any optional "Set" attributes as "Get" only and still be compliant with this object specification. All required attributes must be supported as specified.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-37.2 S-Analog Actuator Object Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Optional | Get | NV | Number of Attributes | USINT | Number of supported attributes | The number of attributes supported by this object instance |
| 2 | Optional | Get | NV | Attribute List | ARRAY OF USINT | List of supported attribute | List of attributes supported by this object instance |
| 3 | Optional | See Semantics | NV | Data Type | USINT | Determines the Data Type of *Value* and all related attributes as specified in this table. | see Semantics section |
| 4 | Optional | See Semantics | NV | Data Units | ENGUNITS | Determines the context of *Value* | see Semantics section |
| 5 | Required | Set | V | Override | USINT | Specifies an override for the physical actuator. For values other than zero (normal control), the *Value* attribute is ignored. | 0 = normal [default]<br>see Semantics section |
| 6 | Required | Set | V | Value | INT or specified by *Data Type* if supported | Analog output value | The uncorrected value. see Semantics section<br>[default] = 0 |
| 7 | Required | Get | V | Status | BYTE | Alarm and Warning State of this object instance | see Semantics section<br>[default] = 0 |
| 8 | Optional | Set | NV | Alarm Enable | BOOL | Enables the setting of the Alarm Bit | 0 = disable [default]<br>1 = enable |
| 9 | Optional | Set | NV | Warning Enable | BOOL | Enables the setting of the Warning Bit | 0 = disable [default]<br>1 = enable |
| 10 | Optional | Set | NV | Offset | INT or specified by *Data Type* if supported | An amount to be added to *Value* prior to the application of gain | see Semantics section<br>0 = [default] |
| 11 | Optional | Set | NV | Bias | INT or specified by *Data Type* if supported | An amount to be added to *Value* after the application of gain | see Semantics section<br>0 = [default] |
| 12 | Required if Attribute “Gain” is other than REAL | Get | NV | Gain Data Type | USINT | Determines the Data Type of attribute *Gain* | see Semantics section |
| 13 | Optional | Set | NV | Gain | REAL or specified by *Gain Data Type* if supported | An amount by which *Value* is scaled prior to driving the physical actuator | see Semantics section<br>1.0 = [default] |
| 14 | Required if Attribute 12 is other than REAL | Get | NV | Unity Gain Reference | REAL or specified by *Gain Data Type* if supported | Specifies the value of the *Gain* attribute equivalent to a gain of 1.0 | Used for normalizing the *Gain* attribute.<br>see Semantics section<br>[default] = 1.0 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Analog Actuator Object, Class Code: $32_{Hex}$**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 15 | Optional | Set | NV | Alarm Trip Point High | INT or specified by *Data Type* if supported | Determines the Value above which an Alarm Condition will occur | see Semantics section<br>[default] = Maximum value for its data type. |
| 16 | Optional | Set | NV | Alarm Trip Point Low | INT or specified by *Data Type* if supported | Determines the Value below which an Alarm Condition will occur | see Semantics section<br>[default] = Minimum value for its data type. |
| 17 | Optional | Set | NV | Alarm Hysteresis | INT or specified by *Data Type* if supported | Determines the amount by which the Value must recover to clear an Alarm Condition | see Semantics section<br>[default] = 0 |
| 18 | Optional | Set | NV | Warning Trip Point High | INT or specified by *Data Type* if supported | Determines the Value above which a Warning Condition will occur | see Semantics section<br>[default] = Maximum value for its data type. |
| 19 | Optional | Set | NV | Warning Trip Point Low | INT or specified by *Data Type* if supported | Determines the Value below which a Warning Condition will occur | see Semantics section<br>[default] = Minimum value for its data type. |
| 20 | Optional | Set | NV | Warning Hysteresis | INT or specified by *Data Type* if supported | Determines the amount by which the Value must recover to clear a Warning Condition | see Semantics section<br>[default] = 0 |
| 21 | Optional | Set | NV | Safe State | USINT | Specifies the behavior of the physical actuator for states other than Execute | see Semantics section<br>0 = [default] |
| 22 | Optional | Set | NV | Safe Value | INT or specified by *Data Type* if supported | The Value to be used for Safe State = Safe Value | see Semantics section<br>0 = [default] |
| 97-98 | Reserved by CIP for Future Use | | | | | | |
| 99 | Conditional * | Get | NV | Subclass | UINT | Identifies a subset of additional instance attributes, services and behaviors. | 0 = No subclass<br>1 – 65535 = Reserved |

* If the value of Subclass is 00, then this attribute is OPTIONAL in implementation, otherwise, this attribute is REQUIRED.

### 5A-37.2.1 Subclasses

Each instance level subclass defines a unique meaning for an overlapping range of instance attribute IDs and/or instance service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given instance is identified by the value of its Subclass instance attribute. The subclasses for this object are specified at the end of this object specification section.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-37.2.2 Semantics:

#### 5A-37.2.2.1 Data Type

All Data Type attributes, including *Data Type* and *Gain Data Type*, use the enumerated values specified in Appendix C-6.1.

The *Data Type* attribute is settable only in the *Idle State* and only if no attribute belonging to the object instance is the endpoint of an I/O connection in the *Established State*.

The *Data Type* attribute may change automatically based upon established I/O connections. See Behavior section for more information on this mechanism.

#### 5A-37.2.2.2 Data Units

Specifies the context of *Value* and related attributes (such as, offset and trip points) for this object instance. See Appendix D for a list of values. A request to set attribute to an unsupported value will return an error response.

The *Data Units* attribute is settable only in the *Idle State*.

#### 5A-37.2.2.3 Value, Offset, Gain, Bias and Unity Gain Reference

The *Offset*, *Gain* and *Bias* attributes are applied to the *Value* attribute to derive the actual signal which drives the physical actuator. The gain is normalized using the *Unity Gain Reference* attribute value. (e.g., for an UINT type *Gain*, a *Unity Gain Reference* value may be 10000, allowing an effective gain of 0.0001 to 6.5535.)

The following formula applies:

physical actuator drive signal = $\boldsymbol{Gain_N} \bullet (\boldsymbol{Value} + \boldsymbol{Offset}) + \boldsymbol{Bias}$

where: $Gain_N$ = Gain / Unity Gain Reference

There may be additional nonlinear conversions applied to the drive signal as specified by the manufacturer.

#### 5A-37.2.2.4 Status

A bit mapped byte which indicates the Alarm and Warning Exception status of the object instance. The following definition applies:

**Table 5A-37.3 Status Attribute**

| Bit | Definition |
|---|---|
| 0 | High Alarm Exception: 0 = cleared; 1 = set |
| 1 | Low Alarm Exception: 0 = cleared; 1 = set |
| 2 | High Warning Exception: 0 = cleared; 1 = set |
| 3 | Low Warning Exception: 0 = cleared; 1 = set |
| 4 | Reserved |
| 5 | Reserved |
| 6 | Reserved |
| 7 | Reserved |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-37.2.2.5 Trip Points and Hysteresis

Trip Point High is the level above which the *Value* attribute will cause an Alarm or Warning exception condition.

Trip Point Low is the level below which the *Value* attribute will cause an Alarm or Warning exception condition.

A *Hysteresis* value specifies the amount by which the *Value* attribute must transition in order to clear an Alarm or Warning condition.

For example: A *Trip Point High* value of 90 and a *Hysteresis* value of 2 will result in an exception condition being set when the *Value* is above 90 and cleared when the *Value* drops below 88. Similarly, A *Trip Point Low* value of 90 and a *Hysteresis* value of 2 will result in an exception condition being set when the *Value* is below 90 and cleared when the *Value* increases above 92.

### 5A-37.2.2.6 Override

This attribute is used to override the function of the *Value* attribute in driving the physical actuator. The primary application of this feature is in devices where the object instance is being driven by another object such as an S-Single Stage Controller object instance.

The *Safe State* attribute provides a mechanism for override depending upon object state and will take precedents over this. That is, if an object instance implements the *Safe State* attribute and related behavior, then this *Override* attribute and related behavior will only function in the Executing State.

**Table 5A-37.4 Override Attribute**

| Attribute Value | State |
|---|---|
| 0 | Normal |
| 1 | Zero |
| 2 | Maximum Value |
| 3 | Hold |
| 4 | Safe State |
| 5-63 | reserved |
| 64-127 | Device Specific |
| 128-255 | Vendor Specific |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5A-37.2.2.7 Safe State

This attribute specifies the behavior of the drive to the physical actuator for states other than Executing. See the S-Device Supervisor object definition in Section 0 for a description of object states. The following values are defined:

**Table 5A-37.5 Safe State Attribute**

| Attribute Value | State |
|---|---|
| 0 | Zero |
| 1 | Maximum Value |
| 2 | Hold Last Value |
| 3 | Use Safe Value |
| 4-63 | Reserved |
| 64-127 | Device Specific |
| 128-255 | Vendor Specific |

#### 5A-37.2.2.8 Safe Value

For *Safe State* set to "Use Safe Value", this attribute holds the value to which the actuator will be driven for object instance states other than Executing. Specifically, this attribute value will become the value of the *Value* attribute. Therefore, the correction formula specified above applies.

### 5A-37.3 Common Services

The S-Analog Actuator Object provides the following Common Services:

**Table 5A-37.6 S-Analog Actuator Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0Ehex | Conditional* | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 10hex | n/a | Required | Set_Attribute_Single | Modifies an attribute value. |
| *The Get_Attribute_Single service is REQUIRED if any attributes are implemented. | | | | |
| See Appendix A for definition of these services | | | | |

### 5A-37.4 Object–Specific Services

The S-Analog Actuator Object provides no Object–Specific services.

### 5A-37.5 Behavior

The behavior of this object is managed by the S-Device Supervisor Object in Section 5A-35.6.

An S-Analog Actuator object instance modifies the *Value* by applying the formula specified above with the associated attribute values. *Value* is specified as *Data Type* and *Data Units*. Optionally, additional corrective algorithms are applied to further correct for various calibration effects. These additional algorithms are specified in other objects, as identified in the device profile, or as extensions, specified by the manufacturer.

All Trip Point calculations, as specified above, utilize the *Value* attribute before the application of *Offset* and *Gain*.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-37.5.1 Data Type

If the implementation of this object specifies more than one valid Data Type value, in the device profile or by vendor, then the following behavior with respect to *Data Type* applies: The Data Type value will be set automatically based upon the first valid I/O connection established by the device. This configuration will then remain in effect for this object instance even after all I/O connections are lost. If, however, a device is specified by a vendor to support only one Data Type, this behavior is not supported.

If no established I/O connections exist, which include an attribute from this object, then the *Data Type* attribute is settable provided that the object is in the *Idle State*.

The following example demonstrates this behavior:

A device profile specifies an instance of the S-Analog Actuator object as well as two static Assembly object instances, both with data attribute components mapped to this object instance. Assembly object instance ID 1 specifies INT data types and Assembly object instance ID 2 specifies REAL data types.

After the device is On-Line, it is configured with an I/O connection to Assembly instance ID 2. When the connection transitions to the *Established State*, this object instance attribute *Data Type* is automatically set with the value for REAL before any data is communicated to, or from, the object instance.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-38 S-Single Stage Controller Object

**Class Code: 33 Hex**

The S-Single Stage Controller Object models a closed-loop control system within a device. Associated with a single stage controller is a Process Variable, a Setpoint and a Control Variable. As normally described by *classic control theory*, a closed-loop controller will drive the Control Variable in order to affect the value of the Process Variable such that it is made to equal the Setpoint. See the Semantics section, below, for more information regarding these variable definitions. Manufacturers may specify additional correction algorithms as extensions to this object.

This object is a member of the *Hierarchy of Semiconductor Equipment Devices*. As such, its behavior is managed by the S-Device Supervisor Object. See Section 0.

## 5A-38.1 Class Attributes

**Table 5A-38.1 S-Single Stage Controller Object Class Attributes**

| Attribute ID | Need in implementation | Access Rule | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | |
| 97-98 | Reserved by CIP for Future Use | | | | |
| 99 | Conditional * | Get | Subclass | UINT | Identifies a subset of additional class attributes, services and behaviors. |
| * If the value of Subclass is 00, which identifies "no subclass", then this attribute is OPTIONAL in implementation, otherwise, this attribute is REQUIRED. | | | | | |

### 5A-38.1.1 Subclasses

Each class level subclass defines a unique meaning for an overlapping range of class attribute IDs and/or class service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given class is identified by the value of its Subclass class attribute. The subclasses for this object are specified at the end of this object specification section.

## 5A-38.2 Instance Attributes

Certain minimal implementations may support any optional “Set” attributes as “Get” only and still be compliant with this object specification. All required attributes must be supported as specified.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-38.2 S-Single Stage Controller Object Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Optional | Get | NV | Number of Attributes | USINT | Number of supported attributes | Number of attributes supported in this object instance |
| 2 | Optional | Get | NV | Attribute List | ARRAY OF USINT | Attribute List | List of attributes supported in this object instance |
| 3 | Optional | See Seman-tics | NV | Data Type | USINT | Determines the Data Type of Setpoint, Process Variable and related attributes | see Semantics section |
| 4 | Optional | See Seman-tics | NV | Data Units | ENGUNITS | Determines the context of the Process related variables such as Setpoint and Process Variable | See Appendix D |
| 5 | Optional | Set | NV | Control Mode | USINT | Specifies the operational mode of the controller | See Semantic section<br>[default] = Normal (0) |
| 6 | Required | Set | V | Setpoint | INT or specified by Data Type if supported | The setpoint to which the process variable will be controlled | See Semantics section.<br>See Behavior section.<br>0 = [default] |
| 7 | Conditional * | Set | V | Process Variable | INT or specified by Data Type if supported | The measured process parameter | The device profile must specify the data connection for this attribute. It may be internally linked to a sensor. See Semantics section.<br>0 = [default] |
| 8 | Optional | Get | NV | CV Data Type | USINT | Determines the Data Type of Control Variable | see Semantics section |
| 9 | Conditional * | Get | V | Control Variable | INT or specified by CV Data Type if supported | The drive signal output of this object. The algorithm by which this attribute is calculated is manufacturer specific. | The device profile must specify the data connection for this attribute. It may be internally linked to an actuator.<br>[default] = 0<br>See Semantics section. |
| 10 | Required | Get | V | Status | BYTE | Alarm and Warning State of this object instance | see Semantics section<br>[default] = 0 |
| 11 | Optional | Set | NV | Alarm Enable | BOOL | Enables the setting of the Alarm Status Bit | 0 = disable [default]<br>1 = enable |
| 12 | Optional | Set | NV | Warning Enable | BOOL | Enables the setting of the Warning Status Bit | 0 = disable [default]<br>1 = enable |
| 13 | Optional | Set | NV | Alarm Settling Time | UINT | Number of Milliseconds allowed for the control-loop to settle to within the error band | see Behavior section<br>[default] = 0 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 14 | Optional | Set | NV | Alarm Error Band | INT or specified by Data Type if supported | The amount by which the Setpoint must equal the Process Variable | see Behavior section<br>[default] = 0 |
| 15 | Optional | Set | NV | Warning Settling Time | UINT | Number of Milliseconds allowed for the control-loop to settle to within the Error Band | see Behavior section<br>[default] = 0 |
| 16 | Optional | Set | NV | Warning Error Band | INT or specified by Data Type if supported | The amount by which the Setpoint must equal the Process Variable | see Behavior section<br>[default] = 0 |
| 17 | Optional | Set | NV | Safe State | USINT | Specifies the Control Variable behavior for states other than Execute | see Semantics section<br>0 = [default] |
| 18 | Optional | Set | NV | Safe Value | INT or specified by Data Type if supported | The value to be used for Safe State = Safe Value | see Semantics section<br>0 = [default] |
| 19 | Optional | Set | NV | Ramp Rate | UDINT | Time in Milliseconds to reach Setpoint | 0 = Disabled [default]<br>x = value in milliseconds<br>see Behavior section |
| 85-96 | Defined by subclasses below. | | | | | | |
| 97-98 | Reserved by CIP for Future Use | | | | | | |
| 99 | Conditional ** | Get | NV | Subclass | UINT | Identifies a subset of additional instance attributes, services and behaviors. | 0 = No subclass<br>1 = PID & Source Select<br>2 = DC Generator<br>3 = RF Generator<br>4 = Frequency Control<br>5 – 65535 = Reserved |

* The Process Variable is only optional if this device includes an internal sensor. Otherwise, the Process Variable is required. Similarly, The Control Variable is only optional if this device includes an internal actuator. Otherwise, the Control Variable is required.

** If the value of Subclass is 00, then this attribute is OPTIONAL in implementation, otherwise, this attribute is REQUIRED.

### 5A-38.2.1 Subclasses

Each instance level subclass defines a unique meaning for an overlapping range of instance attribute IDs and/or instance service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given instance is identified by the value of its Subclass instance attribute. The subclasses for this object are specified at the end of this object specification section.

## 5A-38.3 Semantics:

### 5A-38.3.1 Data Type

All Data Type attributes, including *Data Type* and *CV Data Type*, use the enumerated values specified in Appendix C-6.1.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The *Data Type* attribute is settable only in the *Idle State* and only if no attribute belonging to the object instance is the endpoint of an I/O connection in the *Established State*.

The *Data Type* attribute may change automatically based upon established I/O connections. See Behavior section for more information on this mechanism.

### 5A-38.3.2 Data Units

Specifies the context of *Setpoint and Process Variable* and related attributes (such as, offset and trip points) for this object instance. See Appendix D for a list of values. A request to set attribute to an unsupported value will return an error response.

The *Data Units* attribute is settable only in the *Idle State*.

In applications where this object is used in a relationship with an S-Analog Sensor object, this attribute may be specified as Get only, by the device profile or the vendor, where the value mirrors that of the S-Analog Sensor object *Data Units* attribute.

### 5A-38.3.3 Setpoint, Process Variable and Control Variable

These three attributes compose the primary aspects of basic closed-loop control. The *Process Variable* is the measured parameter of the process or system being controlled. The *Setpoint* is the desired value for the measured parameter. By affecting the value of the *Control Variable*, the closed-loop controller drives the process or system to the desired state of:

Process Variable = Setpoint

The *Control Variable* is, therefore, connected to the process or system in such a way that it affects the value of the *Process Variable*. Examples of *Control Variable / Process Variable* combinations include: heater / temperature; valve / flow; or regulator / pressure.

### 5A-38.3.4 Status

A bit mapped byte which indicates the Alarm and Warning Exception status of the object instance. The following definition applies:

**Table 5A-38.3 Status Attribute**

| Bit | Definition |
|---|---|
| 0 | Alarm Exception: 0 = cleared; 1 = set |
| 1 | Warning Exception: 0 = cleared; 1 = set |
| 2 | Reserved |
| 3 | Reserved |
| 4 | Reserved |
| 5 | Reserved |
| 6 | Reserved |
| 7 | Reserved |

### 5A-38.3.5 Control Mode

This attribute is used to override the value of the *Control Variable* attribute. Further, it may cause the object to modify the internal control algorithm such that a smooth, or “bumpless” transitions occurs upon activating control to setpoint.

The *Safe State* attribute provides a mechanism for override depending upon object state and will take precedents over this. That is, if an object instance implements the *Safe State* attribute and related behavior, then this *Override* attribute and related behavior will only function in the Executing State.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-38.4 Control Mode Attribute**

| Attribute Value | State |
|---|---|
| 0 | Normal |
| 1 | Zero / Off / Closed |
| 2 | Full / On / Open |
| 3 | Hold |
| 4 | Safe State |
| 5-63 | reserved |
| 64-127 | Device Specific (specified by device profile) |
| 128-255 | Vendor Specific |

### 5A-38.3.6 Safe State

This attribute specifies what value will be held in the *Control Variable* attribute for states other than Executing. See the S-Device Supervisor object definition in Section 0 for a description of object states. The following values are defined:

**Table 5A-38.5 Safe State Attribute**

| Attribute Value | State |
|---|---|
| 0 | Zero / Off |
| 1 | Full Scale / On |
| 2 | Hold Last Value |
| 3 | Use Safe Value |
| 4-63 | reserved |
| 64-127 | Device Specific (specified by device profile) |
| 128-255 | Vendor Specific |

### 5A-38.3.7 Safe Value

For Safe State set to Use Safe Value, this attribute holds the value to which the Control Variable attribute will be set for object instance states other than Executing.

## 5A-38.4 Common Services

The S-Single Stage Controller Object provides the following Common Services:

**Table 5A-38.6 S-Single Stage Controller Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0Ehex | Conditional* | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 10hex | n/a | Required | Set_Attribute_Single | Modifies an attribute value. |
| *The Get_Attribute_Single service is REQUIRED if any attributes are implemented. | | | | |
| See Appendix A for definitions of these services | | | | |

## 5A-38.5 Object–specific Services

The S-Single Stage Controller Object provides no Object–Specific services.

## 5A-38.6 Behavior

The behavior of this object is managed by the S-Device Supervisor Object in Section 5A-35.6. Additionally, this object exhibits the following behavior:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-38.6.1 Alarm and Warning Exception Conditions

While in the Executing State as defined by the S-Device Supervisor Object: Immediately upon detecting that the Setpoint does not equal the Process Variable by an amount plus-or-minus the associated (alarm or warning) Error Band, a timer is started. This internal timer is incremented as long as the above condition exists. If the timer exceeds the amount indicated by the associated (alarm or warning) Settling Time and the associated (alarm or warning) Exception Enable is set, then the appropriate (alarm or warning) Exception Condition is set. Note that two internal timers are required in order to support both Alarm and Warning Exception reporting.

This behavior is modified for Ramp Rate values not equal to zero. In such cases, the timer is not enabled until after the expiration of the Ramp Time (see Behavior description below).

### 5A-38.6.2 Ramp Rate

For Ramp Rate values other than zero, the S-Single Stage Controller Object internally modifies the Setpoint value in such a way that the Process Variable is “ramped” to its final value. For example: with a Ramp Rate value of 1000, a new Setpoint value will be internally (transparently) modified, in whatever time increments the object is able to sustain, in order to affect a smooth transition over one second from the old Setpoint to the new Setpoint, finally reaching the new Setpoint at the one second mark.

### 5A-38.6.3 Data Type

If the implementation of this object specifies more than one valid Data Type value, in the device profile or by vendor, then the following behavior with respect to *Data Type* applies: The Data Type value will be set automatically based upon the first valid I/O connection established by the device. This configuration will then remain in effect for this object instance even after all I/O connections are lost. If, however, a device is specified by a vendor to support only one Data Type, this behavior is not supported.

If no established I/O connections exist, which include an attribute from this object, then the *Data Type* attribute is settable provided that the object is in the *Idle State*.

The following example demonstrates this behavior:

A device profile specifies an instance of the S-Single Stage Controller object as well as two static Assembly object instances, both with data attribute components mapped to this object instance. Assembly object instance ID 1 specifies INT data types and Assembly object instance ID 2 specifies REAL data types.

After the device is On-Line, it is configured with an I/O connection to Assembly instance ID 2. When the connection transitions to the *Established State*, this object instance attribute *Data Type* is automatically set with the value for REAL before any data is communicated to, or from, the object instance.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-38.6.4 Control

The application of this object is further specified in the applicable device profile; primarily, the interfaces and object relationships are defined. Generally, the *Process Variable* attribute is restricted to "Get Only" access and an internal connection is defined to another object. Similarly, the *Control Variable* is generally not supported due to internal connections.

When in the EXECUTING state, this object is running an application process designed to cause the *Process Variable* to be driven to the value of the *Setpoint*. In any state other than EXECUTING, the application process is stopped and the *Safe State* is activated for the output of the object.

Any fault detected by the object application process causes the object to transition to the appropriate state as defined by the managing S-Device Supervisor object.

## 5A-38.7 S-Single Stage Controller Object Instance Subclass 01 - (PID and Source Select)

The following specification applies to an instance subclass of this object. The intended applications of this subclass include cases where: PID (Proportion, Integral, Derivative) Servo Control algorithms are utilized and/or more than one Process Variable Source is connected. A typical application for the object instance subclass is that of a Pressure Control Valve.

### 5A-38.7.1 Instance Attributes

The following Instance Attributes are specific for this subclass.

**Table 5A-38.7 PID and Source Select Subclass Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 87 | Conditional * | Get | V | Calibrating State | BOOL | Indicates whether or not the object is in the CALIBRATING state | 0 = Not CALIBRATING<br>1 = CALIBRATING<br>[default] = 0 |
| 88 | Optional | Set | NV | Delay Time | UINT | Defines the time necessary for crossover condition to be present before switching ranges | Time in milliseconds<br>[default] = manufacturer specific<br>see Behavior section |
| 89 | Conditional ** | Set | NV | Sensor Crossover High | INT or specified by Data Type if supported | Defines the crossover point between High to Low range inputs. | [default] = manufacturer specific<br>see Behavior section<br>see Semantics section |
| 90 | Conditional ** | Set | NV | Sensor Crossover Low | INT or specified by Data Type if supported | Defines the crossover point between Low to High range inputs. | [default] = manufacturer specific<br>see Behavior section<br>see Semantics section |
| 91 | Optional | Set | NV | Expected Process Parameter | INT or specified by Data Type is supported | Used to optimize control function | [default] = manufacturer specific |
| 92 | Optional | Set | NV | Kd | REAL | Derivative Gain | [default] = manufacturer specific |
| 93 | Optional | Set | NV | Ki | REAL | Integral Gain | [default] = manufacturer specific |
| 94 | Optional | Set | NV | Kp | REAL | Proportional Gain | [default] = manufacturer specific |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 95 | Optional | Set | NV | Control Direction | BOOL | Defines control direction as Direct or Reverse | 0 = Direct (Downstream Control)<br>1 = Reverse (Upstream Control)<br>[default] = 0<br>see Behavior section |
| 96 | Optional | Set | NV | Process Variable Source | USINT | Source of the Process Variable value for the process control function | 0 = Network<br>1 = S-Analog Sensor object instance #1<br>2 = S-Analog Sensor object instance #2<br>3 = Auto<br>4 = Disable<br>5-255 = Reserved by CIP<br>[default] = 1<br>see Behavior section |

* Required if Calibrating service is supported, otherwise, not required.

** Required if Process Variable Source type 3 is supported, otherwise, not required.

### 5A-38.7.2 Semantics

#### 5A-38.7.2.1 Calibrating State

Indicates the state of the object instance.

#### 5A-38.7.2.2 Delay Time

Specifies the required time of the presence of the Sensor Crossover condition to be true before changing the *Process Variable* source. See Behavior section for more information.

#### 5A-38.7.2.3 Sensor Crossover High, Low

Specifies the value of the *Process Variable*, in percent, that allows automatic switching to the other *Process Variable Sources*. See Behavior section for more information.

#### 5A-38.7.2.4 Expected Process Parameter

Specifies a value for a process parameter that may be applied to the control application process to optimize the control of the *Process Variable*. The context of this attribute is defined by the application. For example, a Pressure Control Valve application may define this attribute to specify the expected Flow Rate through the valve.

#### 5A-38.7.2.5 Kd,Ki,Kp

Specifies the values of a PID control loop. This gives access to adjusting the parameters of the control function for the *Process Variable*.

#### 5A-38.7.2.6 Control Direction

This defines the polarity for control of the process variable. The control mechanism can then be knowledgeable of what effect increasing or decreasing the *Control Variable* will have on the *Process Variable.* See Behavior section for more information.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5A-38.7.2.7 Process Variable Source

The *Process Variable Source* specifies the source of the measured value, or the process variable, being controlled. See Behavior section for more information.

If the Process Variable Source equals zero indicating a Network source, the value of the Process Variable attribute will be updated across a network connection. Sensor Crossover parameters are not used, nor may any automatic switching of the Process Variable Source occur.

If the Process Variable Source equals one or two, the value of the Process Variable attribute is fed by the corresponding S-Analog Sensor object Value attribute. Sensor Crossover parameters are not used nor, may any automatic switching of the Process Variable Source occur.

If the Process Variable Source equals three, the value of the Process Variable attribute is fed by either the low-range or the high-range S-Analog Sensor object Value attribute. In order for three to be a valid attribute value for the Process Variable Source, two S-Analog Sensor instances are required. Selection of the appropriate S-Analog Sensor instance is performed by the device, and defined by the Sensor Crossover High and Sensor Crossover Low attribute values defined below.

Instance 1 for the Process Variable Source S-Analog Sensor is taken as the low range and also is the required instance for direct input when only one Process Variable Source is attached to the device. Instance 2 is taken as the high range, defining this as a higher magnitude above the low range.

**NOTE:** The capability of each device to accurately resolve a particular high range or low range sensors must be specified by the manufacturer. Inappropriate selection of high and low range sensors needs to be addressed in application guides for usage of the device. Care must be given to the compatibility of these ranges to avoid unsatisfactory behavior.

A value of 4 for the Process Variable Source supports device configurations where no process variable exists for this instance. This value prevents action by the controller until configured with a valid Process Variable Source.

### 5A-38.7.3 Services

The following Services are specific for this subclass.

**Table 5A-38.8 PID and Source Select Subclass Services**

| Service | Need in implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| Code | Class | Instance | | |
| 63hex | n/a | Optional | Calibrate | Moves the device to the **Calibrating** state. This service may also end a calibration that is in process. |

#### 5A-38.7.3.1 Calibrate

Used to transition this object to the CALIBRATING substate of the EXECUTING state. Refer to the Behavior section for a description of the CALIBRATING state. This service may also end a calibration procedure that is in process.

**Table 5A-38.9 Calibrate Request Service Data Field Parameters**

| Parameter | Data Type | Description | Semantics of Values |
|---|---|---|---|
| Command | USINT | Specifies whether to start or stop the calibration | 0 = Stop Calibration<br>1 = Start Calibration |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-38.7.4 Behavior

The following additional behavior is specific for this object instance subclass.

**Figure 5A-38.1 Object State Transition Diagram**

**Table 5A-38.10 State Descriptions**

| State | Description |
|---|---|
| CONTROLLING | The object is running an application process designed to position the control valve such that the Process Variable is driven to equal the Setpoint. |
| CALIBRATING | The object is running a manufacturer-specific calibration process. This process may involve an algorithm designed to characterize the system in order to improve the quality of control. |

**Table 5A-38.11 Object State Event Matrix**

| | STATE | |
|---|---|---|
| **EVENT** | **Controlling** | **Calibrating** |
| Start Calibration Request | Transition to CALIBRATING | Error AIRS* |
| Stop Calibration Request | Error AIRS* | Transition to CONTROLLING |
| Calibration Complete | — | Transition to CONTROLLING |
| * Error AIRS = Error Response - Already in Requested Mode/State | | |

#### 5A-38.7.4.1 Process Variable Input Switching

The Sensor Crossover High/Low Variable values for each Process Variable Source defines the value to change to the other Process Variable source. This is only valid with the Process Variable Source Attribute set to automatic (3), which can only be valid with the presence of two Process instances.

#### 5A-38.7.4.2 Delay Time

The Delay Time timer is used to minimize unnecessary switching of the Process Variable Source when more than one source is available and automatic switching is selected. The Delay Time shall be held in reset when the condition for Sensor Crossover is false, and starts when the condition is true. After starting to count, the Delay Time resets anytime the condition is not true. The Delay Time also resets after a switch has occurred.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-38.7.4.3 Sensor Crossover Low

This parameter is only used when there are two instances of the S-Analog Object for Process Variable Sources and the value of the Process Variable Source is 3. It specifies the value of the Process Variable that allows the device to switch automatically from the Process Variable Source instance 1 to instance 2. Its value is valid only in the range of the instance 1 S-Analog Sensor. To rephrase this, this variable describes a value when to automatically switch from the low to high range of Process Variable Sources. Specifying a value outside the range of the low range sensor is not allowed.

### 5A-38.7.4.4 Sensor Crossover High

This parameter only used when there are two instances of the S-Analog Object for Process Variable Sources and the value of the Process Variable Source is 3. It specifies the value of the Process Variable that allows automatic switching from the Process Variable Source instance 2 to instance 1. Its value is valid only in the range of the instance 2 S-Analog Sensor. To rephrase this, this variable describes a value when to automatically switch from the high to low range of Process Variable Sources. Specifying a value outside the range of the high range sensor is not allowed.

### 5A-38.7.4.5 Control Direction

*Direct-Acting Controller* — as the *Process Variable* increases, the *Control Variable* increases (and visa versa). For example, where this object instance subclass is applied to a variable conductance valve downstream of a pressure control process (downstream pressure control); an increase in pressure (*Process Variable*) causes an increase in valve opening (*Control Variable*) resulting in a commensurate decrease in pressure, whereby, the pressure is maintained at *Setpoint*.

*Reverse-Acting Controller* — as the *Process Variable* increases, *Control Variable* decreases (and visa versa). Referencing the above example (*Direct-Acting Controller*), this configuration would apply to a variable conductance valve upstream of a pressure control process (upstream pressure control).

### 5A-38.7.4.6 Process Variable Source

If the *Process Variable Source* equals zero indicating a Network source, the value of the *Process Variable* attribute will be updated across a network connection. Sensor Crossover parameters are not used, nor may any automatic switching of the Process Variable Source occur.

If the *Process Variable Source* equals one or two, the value of the *Process Variable* attribute is fed by the corresponding S-Analog Sensor object *Value* attribute. Sensor Crossover parameters are not used nor, may any automatic switching of the Process Variable Source occur.

If the *Process Variable Source* equals three, the value of the *Process Variable* attribute is fed by either the low-range or the high-range S-Analog Sensor object *Value* attribute. In order for three to be a valid attribute value for the *Process Variable Source*, two S-Analog Sensor instances are required. Selection of the appropriate S-Analog Sensor instance is performed by the device, and defined by the *Sensor Crossover High and Sensor Crossover Low* attribute values defined below.

Instance 1 for the *Process Variable Source* S-Analog Sensor is taken as the low range and also is the required instance for direct input when only one *Process Variable Source* is attached to the device. Instance 2 is taken as the high range, defining this as a higher magnitude above the low range.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**NOTE:** The capability of each device to accurately resolve a particular high range or low range sensors must be specified by the manufacturer. Inappropriate selection of high and low range sensors needs to be addressed in application guides for usage of the device. Care must be given to the compatibility of these ranges to avoid unsatisfactory behavior.

A value of 4 for the *Process Variable Source* supports device configurations where no process variable exists for this instance. This value prevents action by the controller until configured with a valid *Process Variable Source.*

## 5A-38.8 S-Single Stage Controller Instance Subclass 02 - DC Generator

The following specification applies to a subclass of this object for application in DC Power Generator devices such as those used to create plasma energy.

### 5A-38.8.1 Instance Attributes

The following Instance Attributes are specified for this object subclass.

**Table 5A-38.12 DC Generator Subclass Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 85 | Optional | Get | NV | Output Max | DINT | Factory set Device Capability<br>see Below | Watts, Volts or milliAmps as Specified by Data Units attribute (ID 4) |
| 86 | Optional | Set | NV | Output Limit | DINT | Desired limit of the Device<br>see Below | Watts, Volts or milliAmps as Specified by Data Units attribute (ID 4) |
| 87 | Optional | Set | NV | Arc Trip Level | DINT | Output voltage level which determines an arc condition | Volts |
| 88 | Optional | Set | NV | Arc Count Limit | DINT | Number of arcs that create an Arc Limit Exceeded Warning Condition | Counts<br>[default] = 1 |
| 89 | Optional | Get | V | Arc Counter | UDINT | Number of arcs detected since Output Power was last started | Counts |
| 90 | Conditional* | Set | NV | Pulse Enable | BOOL | Enables the Pulse Application Process<br>See Below | 0 = Disable [default]<br>1 = Enable |
| 91 | Conditional* | Set | NV | Pulse Second Setpoint | INT or specified by Data Type if supported | The setpoint value for the second half of a pulse period | Watts, Volts or milliAmps as Specified by Data Units attribute (ID 4)<br>[default] = 0 |
| 92 | Conditional* | Set | NV | Pulse Period Duration | DINT | The duration of one complete on/off cycle | microseconds |
| 93 | Conditional* | Set | NV | Pulse Duty Cycle | UINT | The percentage of the Pulse Period for which the setpoint is at its normal value | % |
| 94 | Optional | Set | NV | Sync Enable | BOOL | Enables an external synchronizing signal to begin a Pulse Period | 0 = Disabled [default]<br>1 = Enabled |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 95 | Optional | Set | NV | Sync Phase Delay | DINT | Specifies a delay between the sync signal and beginning of the Pulse Period | microseconds |
| 96 | Optional | Set | NV | Ramp Rate Increment | UINT | Specifies a setpoint ramp as described below | Range = 0 to 4095<br>0 - 100% Setpoint per Second<br>0 = Disabled [default] |

* Required if Pulsed Power Delivery is supported

### 5A-38.8.2 Semantics

#### 5A-38.8.2.1 Output Max

This is a factory set specification of the capability of the device. The context of the attribute is determined by the value of the *Data Units* attribute (ID 4). Depending upon the use of this Object Instance, this value specifies the maximum, Output Power, Output Voltage or Output Current.

#### 5A-38.8.2.2 Output Limit

This attribute is set by the user to limit the output. The device shall not deliver any output beyond this value. The context of the attribute is the same as that of the *Output Max* attribute above. A request to set this attribute beyond the value of *Output Max* returns an Invalid Attribute Value error.

#### 5A-38.8.2.3 Ramp Rate Increment

Upon receiving a new setpoint value for output power, the S-Single Stage Controller Object instance will internally ramp to the new setpoint value as specified by the *Ramp Rate Increment* attribute. For a *Ramp Rate Increment* value of zero, the ramping is disabled and the new setpoint takes effect immediately or as determined by the time based *Ramp Rate* attribute (ID 19) value. Otherwise, the *Ramp Rate Increment* behavior overrides that of the time based *Ramp Rate*.

For values in the range of 1-4095, corresponding to the range of $2.442 \times 10^{-4}$ % to 100 %, the setpoint is ramped at a rate equal to the percentage of the difference between the initial setpoint and the new setpoint per second (e.g., 50% specifies half the difference).

For example: Ramp Rate = 2047 (50%); initial setpoint = 75% of Full Scale; new setpoint = 100% of Full Scale. Results: effective setpoint ramps from 75% to 100% of Full Scale at the rate of 12.5% of Full Scale per second and reaching final value in 2 seconds.

### 5A-38.8.3 Services

There are no additional services defined for this subclass

### 5A-38.8.4 Behavior

#### 5A-38.8.4.1 Pulse Application Process

The Pulse Application Process causes the S-Single Stage Controller Object Instance to behave as though its setpoint attribute is periodically switching between two values. The first value is that which is set to the Setpoint attribute. The second value is that which is set to the Pulse Second Setpoint attribute.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-38.9 S-Single Stage Controller Instance Subclass 03 - RF Generator

The following specification applies to a subclass of this object for application in RF Power Generator devices such as those used to create plasma energy.

### 5A-38.9.1 Instance Attributes

The following Instance Attributes are specified for this object subclass.

**Table 5A-38.13 RF Generator Subclass Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 88 | Optional | Get | NV | Output Max | DINT | Factory set Device Capability<br>see Below | Watts, milliWatts, Volts or milliAmps as Specified by Data Units attribute (ID 4) |
| 89 | Optional | Set | NV | Output Limit | DINT | Desired limit of the Device<br>see Below | Watts, milliWatts, Volts or milliAmps as Specified by Data Units attribute (ID 4) |
| 90 | Conditional* | Set | NV | Pulse Enable | BOOL | Enables the Pulse Application Process | 0 = Disable [default]<br>1 = Enable |
| 91 | Conditional* | Set | NV | Pulse Second Setpoint | DINT | The setpoint value for the second half of a pulse period | Watts, milliWatts, Volts or milliAmps as Specified by Data Units attribute (ID 4)<br>[default] = 0 |
| 92 | Conditional* | Set | NV | Pulse Period Duration | DINT | The duration of one complete on/off cycle | microseconds |
| 93 | Conditional* | Set | NV | Pulse Duty Cycle | UINT | The percentage of the Pulse Period for which the setpoint is at its normal value | % |
| 94 | Optional | Set | NV | Sync Enable | BOOL | Enables an external synchronizing signal to begin a Pulse Period | 0 = Disabled [default]<br>1 = Enabled |
| 95 | Optional | Set | NV | Sync Phase Delay | DINT | Specifies a delay between the sync signal and beginning of the Pulse Period | microseconds |
| 96 | Optional | Set | NV | Ramp Rate Increment | UINT | Specifies a setpoint ramp as described below | Range = 0 to 4095<br>0 - 100% Setpoint per Second<br>0 = Disabled [default] |

* Required if Energy Control is supported
** Required if Target Life tracking is supported
*** Required if Pulsed Power Delivery is supported

### 5A-38.9.2 Semantics

#### 5A-38.9.2.1 Output Max

This is a Factory set specification of the capability of the device. The context of the attribute is determined by the value of the *Data Units* attribute (ID 4). Depending upon the use of this Object Instance, this value specifies the maximum, Output Power or Delivered Power.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5A-38.9.2.2 Output Limit

This attribute sets the limit beyond which the device will not operate. The context of the attribute is the same as that of the *Output Max* attribute above. A request to set this attribute beyond the value of *Output Max* returns an Invalid Attribute Value error.

#### 5A-38.9.2.3 Ramp Rate Increment

Upon receiving a new setpoint value for output power, the S-Single Stage Controller Object instance will internally ramp to the new setpoint value as specified by the *Ramp Rate Increment* attribute. For a *Ramp Rate Increment* value of zero, the ramping is disabled and the new setpoint takes effect immediately or as determined by the time based *Ramp Rate* attribute (ID 19) value. Otherwise, the Ramp Rate Increment behavior overrides that of the time based Ramp Rate.

For values in the range of 1-4095, corresponding to the range of $2.442 \times 10^{-4}$ % to 100 %, the setpoint is ramped at a rate equal to the percentage of the difference between the initial setpoint and the new setpoint per second (e.g., 50% specifies half the difference).

For example: Ramp Rate = 2047 (50%); initial setpoint = 75% of Full Scale; new setpoint = 100% of Full Scale. Results: effective setpoint ramps from 75% to 100% of Full Scale at the rate of 12.5% of Full Scale per second and reaching final value in 2 seconds.

### 5A-38.9.3 Services

There are no additional services defined for this subclass.

### 5A-38.9.4 Behavior

#### 5A-38.9.4.1 Pulse Application Process

The Pulse Application Process causes the S-Single Stage Controller Object Instance to behave as though its setpoint attribute is periodically switching between two values. The first value is that which is set to the Setpoint attribute. The second value is that which is set to the Pulse Second Setpoint attribute.

## 5A-38.10 S-Single Stage Controller Instance Subclass 04 - Frequency Control

The following specification applies to a subclass of this object for application in RF Power Generator devices such as those used to create plasma energy.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-38.10.1 Instance Attributes

The following Instance Attributes are specified for this object subclass.

**Table 5A-38.14 Frequency Control Subclass Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 92 | Optional | Get | NV | Minimum Frequency | DINT | Factory Set | Hertz |
| 93 | Optional | Get | NV | Maximum Frequency | DINT | Factory Set | Hertz |
| 94 | Optional | Set | NV | Default Frequency | DINT | Factory Set | Hertz |
| 95 | Optional | Set | NV | Frequency Limit Low | DINT | Limits the Output Frequency of the device | Hertz<br>May not be set beyond Min Frequency |
| 96 | Optional | Set | NV | Frequency Limit High | DINT | Limits the Output Frequency of the device | Hertz<br>May not be set beyond Max Frequency |

### 5A-38.10.2 Semantics

#### 5A-38.10.2.1 Minimum, Maximum and Default Frequency Attributes

*Minimum, Maximum and Default Frequency* attributes specify the capabilities of the device.

#### 5A-38.10.2.2 Frequency Limit High and Low Attributes

*Limit* attributes allow the user to set bounds beyond which the device will not operate. A request to set a *limit* attribute or the *default* attribute beyond the bounds of a specified *Min or Max* value will return an Invalid Attribute Value error.

#### 5A-38.10.2.3 Setpoint

A Setpoint value of zero causes the S-Single Stage Controller object instance to perform the Auto-tune Application Process. Any other value of setpoint (within specified limits) causes the S-Single Stage Controller object instance to perform the Frequency Control Application Process.

Data Type = DINT

Data Units = Hertz

### 5A-38.10.3 Services

There are no additional services defined for this subclass.

### 5A-38.10.4 Behavior

#### 5A-38.10.4.1 Frequency Control Application Process

This application process controls the output frequency to the set value as specified by the Setpoint attribute of this object instance.

#### 5A-38.10.4.2 Auto-tune Application Process

This application process controls the output frequency in order to affect optimum power delivery efficiency in RF Power Generator devices. The vendor specific algorithm tunes the output frequency in order to minimize Reflected Power for the device's application.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-39 S-Gas Calibration Object

**Class Code: 34Hex**

An S-Gas Calibration Object affects the behavior of an associated S-Analog Sensor object instance; a device profile will show a relationship between these two objects where an S-Gas Calibration Object is used. The S-Analog Sensor object uses a selection attribute as the gas type selection mechanism. The S-Gas Calibration Object provides the data with which a device enacts the appropriate calibration algorithm for a given gas type. Each S-Gas Calibration Object Instance contains a set of attribute values for one particular calibration set; each identified by the Gas Standard Number.

The S-Gas Calibration class level object provides a service for retrieving a list of all valid object instances. The service response includes a list of elements. Each element includes: Instance ID, Gas Standard Number and the valid S-Analog Sensor object instance ID for which the instance is valid.

There may be more than one instance with the same Gas Standard Number. These instances may be differentiated by Full Scale, Gas Symbol, Additional Scaler and/or other parametric distinctions, including valid S-Analog Sensor object instance ID. The distinctions may, or may not, be evident in the Get_All_Instances service response, depending upon what the distinction is.

S-Gas Calibration Objects most often utilize the region of Manufacturer Specified Attributes (ID > 100) for specific calibration parameters.

This object is a member of the *Hierarchy of Semiconductor Equipment Devices*. As such, its behavior is managed by the S-Device Supervisor Object. See Section 0.

The S-Gas Calibration object makes use of a list of Standard Gas Type Numbers. This list is described in publication:

SEMI E52-95 “Practice for Referencing Gases Used in Digital Mass Flow Controllers”, Semiconductor Equipment and Materials International (SEMI), Mountain View, CA 94043-4080.

NOTE: It is implied that the reference above is to the latest revision as specified by SEMI.

## 5A-39.1 Class Attributes

**Table 5A-39.1 S-Gas Calibration Object Class Attributes**

| Attribute ID | Need in implementation | Access Rule | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | |
| 97-98 | Reserved by CIP for Future Use | | | | |
| 99 | Conditional * | Get | Subclass | UINT | Identifies a subset of additional class attributes, services and behaviors. |
| * If the value of Subclass is 00, which identifies "no subclass", then this attribute is OPTIONAL in implementation, otherwise, this attribute is REQUIRED. | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-39.1.1 Subclasses

Each class level subclass defines a unique meaning for an overlapping range of class attribute IDs and/or class service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given class is identified by the value of its Subclass class attribute. The subclasses for this object are specified at the end of this object specification section.

## 5A-39.2 Instance Attributes

Certain minimal implementations may support any optional "Set" attributes as "Get" only and still be compliant with this object specification. All required attributes must be supported as specified.

**Table 5A-39.2 S-Gas Calibration Object Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Optional | Get | NV | Number of Attributes | USINT | Number of attributes supported | |
| 2 | Optional | Get | NV | Attribute List | ARRAY OF USINT | List of attributes supported by this object instance | |
| 3 | Required | Get | NV | Gas Standard Number | UINT | Gas Type Number | [default] = 0 (no gas type specified)<br>see Semantics section |
| 4 | Required | Get | NV | Valid Sensor Instance | UINT | S-Analog Sensor object instance ID for which this object instance is valid | 0 = No Valid Sensor<br>n = Instance ID<br>see Semantics section<br>[default] = 0 |
| 5 | Optional | Set | NV | Gas Symbol | SHORT STRING | Gas Type Name | see Semantics section<br>[default] = null |
| 6 | Optional | Set * | NV | Full Scale | STRUCT of: | Full Scale of the device using this object instance | see Semantics section<br>[default] = 0, 0 |
| | | | | | REAL | Amount | The amount of measured parameter corresponding to full scale. |
| | | | | | ENGUNITS | Units | The units for the above. See Data Units Appendix D |
| 7 | Optional | Set | NV | Additional Scaler | REAL | Additional Correction Factor | In addition to the correction algorithm, this amount is multiplied to the reading. Generally used for Gas Correction for a gas other than the type identified for the object instance by attribute 3.<br>(e.g., scale a nitrogen object instance to measure argon).<br>Default = 1.0 |
| 8 | Optional | Get | NV | Calibration Date | DATE | Date of Calibration | The date this object instance was last calibrated<br>[default] = 0 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**S-Gas Calibration Object, Class Code: $34_{Hex}$**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 9 | Optional | Get | NV | Calibration Gas Number | UINT | Calibration Gas | The gas number of the gas used to calibrate this object instance.<br>[default] = 0 |
| 10 | Optional | Get | NV | Gas Correction Factor | REAL | Gas Correction Factor<br>For devices that support simple correction factors (as opposed to algorithms) for gas selection. | [default] = 1.0 |
| 95-96 | Defined by Subclasses below | | | | | | |
| 97-98 | Reserved by CIP for Future Use | | | | | | |
| 99 | Conditional ** | Get | NV | Subclass | UINT | Identifies a subset of additional instance attributes, services and behaviors. | 0 = No subclass<br>1 = Standard T & P<br>2 – 65535 = Reserved |

* If this attribute is supported, Set Access is optional.

** If the value of Subclass is 00, then this attribute is OPTIONAL in implementation, otherwise, this attribute is REQUIRED.

### 5A-39.2.1 Subclasses

Each instance level subclass defines a unique meaning for an overlapping range of instance attribute IDs and/or instance service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given instance is identified by the value of its Subclass instance attribute. The subclasses for this object are specified at the end of this object specification section.

## 5A-39.3 Semantics:

### 5A-39.3.1 Gas Standard Number

Used to identify a gas standard number, for which the object instance is currently calibrated. See Instance Application Example below.

The actual coding of the values are described in the following publication:

See introduction section (above) for reference to the SEMI publication: “Practice for Referencing Gases Used in Digital Mass Flow Controllers”.

Since the actual attributes, and their context, for the parameterization of object instances for particular gas types is beyond the scope of this standard (i.e., vendor specific) the Access Rule for this attribute has been specified as Get. Vendors may choose to specify an Access Rule of Set for this attribute.

### 5A-39.3.2 Valid Sensor Instances

This attribute specifies the S-Analog Sensor object instance for which the S-Gas Calibration object instance is valid. An S-Gas Calibration object instance will be valid for zero or one S-Analog Sensor object instances.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-39.3.3 Gas Symbol

This optional attribute is a string coded representation of the name of the gas for which the object instance has been configured. It is coded as a user defined text symbol or it is coded as defined in the above referenced SEMI publication.

This attribute may indicate a different gas from the one which has been specified by the *Gas Standard Number*. See Instance Application Example below.

### 5A-39.3.4 Full Scale

This optional attribute identifies the amount of measured parameter (e.g., Mass Flow) corresponding to the Full Scale of the associated S-Analog Sensor object. A primary purpose for this attribute is to allow for simple S-Analog Sensor object implementations where the Value is reported in raw units; this attribute allows a mapping to engineering units.

For example, the Full Scale for a S-Gas Calibration object may be 100 sccm, while the Full Scale for the associated S-Analog Sensor object may be 20,000 counts (i.e., S-Analog Sensor object Data Type = INT and Data Units = Counts).

### 5A-39.3.5 Instance Application Example

The following is an example to demonstrate the usage of Gas Calibration object instances and their attributes:

A device has been supplied with three gas calibration object instances: nitrogen (13)[2], helium (1) and argon (4). The user wishes to use the device for silane (39) and knows that a correction factor of 0.60 will properly convert a nitrogen calibration for this application. The object instance for nitrogen would be selected and the *Additional Scaler* attribute for this instance would be set to 0.60. To identify this modification, the *Gas Symbol* may be set to read “silane”, “SiH4”, or “39”.

## 5A-39.4 Common Services

The S-Gas Calibration Object provides the following Common Services:

**Table 5A-39.3 S-Gas Calibration Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0Ehex | Required | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 10hex | Required | Required | Set_Attribute_Single | Modifies an attribute value. |
| See Appendix A for a description of these services | | | | |

## 5A-39.5 Object–specific Services

**Table 5A-39.4 S-Gas Calibration Object-specific Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 4Bhex | Required | n/a | Get_All_Instances | Requests a list of all available object instances with their respective gas numbers |

2 The number used with element names shown here is the *Gas Standard Number*


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-39.5 Success Response Service Data Field Parameters**

| Parameter | Required | Data Type | Description | Semantics of Values |
|---|---|---|---|---|
| Size of List | Required | UINT | Specifies the number of elements in the Array | Number of gas calibrations in the list |
| List of Gas Calibrations | Required if Size > 0 | ARRAY of | Supported List | The list of gas calibrations |
| | | STRUCT of | Supported Gas Type | |
| | | UINT | S-Gas Calibration Object Instance ID | |
| | | UINT | Gas Standard Number | |
| | | UINT | Valid Sensor Instance | |

## 5A-39.6 S-Gas Calibration Object Behavior

The behavior of this object is managed by the Device Supervisor Object, defined in Section 5A-35.6.

## 5A-39.7 S-Gas Calibration Object Instance Subclass 01 (Standard T & P)

The following specification applies to a subclass of this object for application in Mass Flow Controller devices.

### 5A-39.7.1 Instance Attributes

The following Instance Attributes are specified for this object subclass 01.

**Table 5A-39.6 Standard T & P Subclass Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 95 | Optional | Get | Calibration Pressure | REAL | The gas pressure in KiloPascal | The Standard Pressure with respect to the calibration conditions.<br>Default = 101.32 |
| 96 | Optional | Get | Calibration Temperature | REAL | The Gas Temperature in Degrees C | The Standard Temperature with respect to the calibration conditions.<br>Default = 0.0 |

## 5A-39.8 Services

There are no additions or restrictions to the Object Services for this object subclass.

## 5A-39.9 Behavior

There are no additions or restrictions to the Behavior for this object subclass.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-40 Trip Point Object

## Class Code: 35Hex

The Trip Point Object models the action of trip points for a device, often corresponding to physical outputs (Discrete Output Object). Each Trip Point instance has a source pointer (an attribute of data type Packed EPATH referencing an input value) and a destination pointer (an attribute of data type Packed EPATH referencing a discrete output value). A trip point value, designated as a High or Low trip point, is compared to the specified source value. This trip point is intended to be used as a process control indicator.

**Table 5A-40.1 Revision History**

| Revision | Description |
|---|---|
| 1 | Revision 1 of this object is deprecated. Revision 1 definition of this object can be found in edition 3.7 of this specification. |
| 2 | Added instance attributes: Trip Point High Hysteresis, Trip Point Low Hysteresis and Trip Status<br>Changed the default states of the instance attributes: High Trip Point, High Trip Enable, Low Trip Point, Low Trip Enable, Hysteresis, Trip Point High Hysteresis, Trip Point Low Hysteresis<br>Status renamed to Output Status |

## 5A-40.1 Class Attributes

**Table 5A-40.2 Trip Point Object Class Attributes**

| Attribute ID | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Required | Get | Revision | UINT | Revision of this object | The current value assigned to this attribute is two (02). |
| 2 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | |
| 97 - 98 | Reserved by CIP | | | | | |
| 99 | Conditional * | Get | Subclass | UINT | Identifies a subset of additional attributes, services and behaviors. The subclasses for an object are specified at the end of the object specification section. | |
| * If the value of Subclass is 00, then this attribute is OPTIONAL in implementation, otherwise, this attribute is REQUIRED. | | | | | | |

### 5A-40.1.1 Subclasses

Each class level subclass defines a unique meaning for an overlapping range of class attribute IDs and/or class service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given class is identified by the value of its Subclass class attribute. The subclasses for this object are specified at the end of this object specification section.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-40.2 Instance Attributes

**Table 5A-40.3 Trip Point Object Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Optional | Get | NV | Number of Attributes | USINT | Number of attributes supported | |
| 2 | Optional | Get | NV | Attribute List | ARRAY OF USINT | List of attributes supported by this object instance | |
| 3 | Conditional<br>[At least one of attributes 3 or 5 are required.] | Set | NV | High Trip Point | INT or based Data Type attribute, if supported. | Defines the Value at or above which a trip point condition will occur | [default = defined by vendor]<br>See Semantics section |
| 4 | Conditional<br>[Required if attr 3 "High Trip Point" exists] | Set | NV | High Trip Enable | BOOL | Enables the High Trip Point setting. | [default = defined by vendor]<br>See Semantics section |
| 5 | Conditional<br>[At least one of attributes 3 or 5 are required.] | Set | NV | Low Trip Point | INT or based Data Type attribute, if supported. | Defines the Value at or below which a trip point condition will occur | [default = defined by vendor]<br>See Semantics section |
| 6 | Conditional<br>[Required if attr.5 “Low Trip Point” exists] | Set | NV | Low Trip Enable | BOOL | Enables the Low Trip Point setting. | [default = defined by vendor]<br>See Semantics section |
| 7 | Required | Get | V | Output Status | BOOL | State of this object instance | 0 = trip point condition does not exist (unasserted)<br>1 = trip point condition exists (asserted)<br><br>See Semantics section |
| 8 | Optional | Set | NV | Polarity | BOOL | Polarity of Output as derived to Status. | 0 = Normal (Output = Status)<br>1 = Reverse (Output = Status inverted)<br>See Semantics section |
| 9 | Optional | Set | NV | Override | USINT | Specifies an override Status.<br>Note: This attribute may also be set internally by the device during different States. | 0 = Normal<br>1 = Force FALSE or unasserted<br>2 = Force TRUE or asserted<br>3 = Freeze Status and Output at existing values<br>4 – 255 = Reserved by CIP |
| 10 | Conditional<br>See semantics 5A-40.3.2 | Set | NV | Hysteresis | INT or based on Data Tpe attribute if supported | Determines the amount by which the Input must recover to clear a trip point high and/or low condition | See Semantics section 5A-40.3.2<br>[default = defined by vendor]<br>A negative value for Hysterisis is not allowed |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Trip Point Object, Class Code: $35_{Hex}$

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 11 | Optional | Set | NV | Delay | UINT | Specifies the amount of time a trip condition must exist before it is reported to Status | Time in milliseconds<br>See Semantics section 5A-40.3.2<br>[default=0] |
| 12 | Required | Set | NV | Destination | Packed EPATH | Specifies the path of the destination attribute whose value will be set by Output | See Semantics |
| 13 | Required | Get | V | Output | BOOL | Output of the object the value of which is sent to destination | = Status as a function of Polarity<br>See Semantics |
| 14 | Required | Set | NV | Source | Packed EPATH | Specifies the path of the source attribute whose value is retrieved for Input | See Semantics |
| 15 | Required | Set | V | Input | INT or specified by Data Type | Input to the object whose value is retrieved from source | See Semantics |
| 16 | Optional | Get | NV | Data Units | ENGUNITS | Units of Input, Trip Point, Hysteresis, etc. | See Semantics |
| 17 | Optional | Get | NV | Data Type | USINT | Data Type of Input, Trip Point, Hyseresis, etc. | [default] = INT<br>See Semantics |
| 18 | Conditional<br>See semantics 5A-40.3.2 | Set | NV | Trip Point High Hysteresis | INT or<br>based on Data Type attribute, if supported. | Determines the amount by which the Input must recover to clear a trip point high condition | See Semantics section 5A-40.3.2<br>[default = defined by vendor]<br>A negative value is not allowed |
| 19 | Conditional<br>See semantics 5A-40.3.2 | Set | NV | Trip Point Low Hysteresis | INT or<br>based on Data Type attribute, if supported. | Determines the amount by which the Input must recover to clear a trip point low condition | See Semantics section 5A-40.3.2<br>[default = defined by vendor]<br>A negative value is not allowed |
| 20 | Conditional<br>Needs to be implemented if the device supports a High Trip Point AND a Low Trip Point | Get | V | Trip Status | USINT | Describes the activator for the trip point condition.<br>Activator may be a<br>High Trip Point condition<br>and/or a<br>Low Trip Point condition. | 0 = no trip point condition exists<br>1 = low trip point condition exists<br>2 = high trip point condition exists<br>3 = high trip and low trip condition exist<br>4-255: reserved |
| 97 - 98 | Reserved by CIP | | | | | | |
| 99 | Conditional | Get | NV | Subclass | UINT | Identifies a subset of additional attributes, services and behaviors. The subclasses for this object are specified at the end of this object specification section. | 0 = No subclass<br>n = subclass as defined herein |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-40.2.1 Subclasses

Each instance level subclass defines a unique meaning for an overlapping range of instance attribute IDs and/or instance service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given instance is identified by the value of its Subclass instance attribute. The subclasses for this object are specified at the end of this object specification section.

## 5A-40.3 Semantics

### 5A-40.3.1 High/Low Trip Point, Enable and Output Status

*High* or *Low Trip Point* is compared to the *Input* value to generate a trip point condition.

If a single trip point for this instance is required, either the High or Low Trip Point may be enabled, determining not only the direction of the trip point, but also the direction of hysteresis. If two or more separate trip points are required, each exhibiting a separate *Status*, then two or more instances of this object are required.

A trip region can be established with only one instance by enabling both High and Low settings. *Output Status* will be set if the input value is at or below the *Low Trip Point* OR at or above the *High Trip Point*, cleared if the input value lies between the trip points.

### 5A-40.3.2 Hysteresis and Delay

The Delay value specifies how long the trip condition must exist or clear before it is reported in Output Status. The Hysteresis value specifies the amount by which the Input value must transition in order to clear a trip point condition. Accordingly, Hysteresis can be substituted by Trip Point High Hysteresis or Trip Point Low Hysteresis depending on the attribute used in the implementation.

The following relationships demonstrate the logic for a High Trip Point with Hysteresis and assume the use of an internal Timer:

For *Status* not set:
If (*Input* >= *Trip Point High*) and (Timer not running)
    Then start a Timer

If (*Input* >= *Trip Point High*) and (Time >= *Delay*)
    Then set *Output Status*
If (*Input* < *Trip Point High - Hysteresis*) and (Time < *Delay*)
    Then reset Timer

For *Status* set:
If (*Input* < *Trip Point High - Hysteresis*) and (Timer not running)
    Then start a Timer
If (*Input* < *Trip Point High - Hysteresis*) and (Time >= *Delay*)
    Then clear *Output Status*
If (*Input* >= *Trip Point High*) and (Time < *Delay*)
    Then clear *Output Status*

Example: *High Trip Point* = 100, *Hysteresis* = 2, and *Delay* = 1000: will result in a trip point condition being set when the *Input* stays at or above 100 for 1 second and cleared when *Input* drops below 98 for 1 second. Similarly: *Low Trip Point* = 100, *Hysteresis* = 2, and a *Delay* = 1000: will result in a trip point condition being set when *Input* falls at or below 100 for 1 second and cleared when *Input* increases above 102 for 1 second.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Attributes 10, 18 and 19 are conditional.

If attribute 10 is implemented then neither attributes 18 nor 19 shall be implemented. If attribute 3 is not implemented, then attribute 18 shall not be implemented. If attribute 5 is not implemented, then attribute 19 shall not be implemented.

The object implementation must support a hysteresis attribute. If attribute 3 (High Trip Point) is implemented, then either attribute 10 or 18 shall be implemented. If attribute 5 (Low Trip Point) is implemented, then either attribute 10 or 19 shall be implemented.

The possible implementations of the different types of hysteresis are shown in the following table.

**Table 5A-40.4 Allowed implementations of the different types of hysteresis**

| | | Device Implementation | | | | |
|---|---|---|---|---|---|---|
| | | **High Trip supported only**<br><br>**Recommended for new design** | **High Trip supported only** | **Low Trip supported only**<br><br>**Recommended for new design** | **Low Trip supported only** | **High Trip and Low Trip supported** |
| **Attributes to be implemented** | Attr. 10 Hysteresis | | X | | X | |
| | Attr. 18 Trip Point High Hysteresis | X | | | | X |
| | Attr. 19 Trip Point Low Hysteresis | | | X | | X |

### 5A-40.3.3 Data Type and Data Units

Specifies the context of *Input* and related attributes (such as *Hysteresis*) for this object instance. Data Units and Data Type will match the source attribute's Data Units and Data Type. See Appendix C for Data Type definitions and Appendix D for Data Units (ENGUNITS) definition.

### 5A-40.3.4 Polarity and Output

The value of the *Output* attribute is derived by combining the value of the *Output Status* attribute with that of the *Polarity* attribute. For a *Polarity* value of Normal, the *Output* will equal the value of *Output Status*. For a *Polarity* value of Reverse, the *Output* will equal the value of *Output Status* inverted. The following relationships demonstrate this logic:

For (Polarity = 0): Output = Output Status

For (Polarity = 1): Output = Output Status Inverted

The *Polarity* attribute is optional and is anticipated to be used only to overcome a hardware limitation.

### 5A-40.3.5 Source and Input

The *Source* attribute defined as data type Packed EPATH (see Appendix C, Table C-1.3), specifies the path for the input whose value is retrieved and becomes the value of *Input*.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-40.3.6 Destination and Output

The *Destination* attribute, defined as data type Packed EPATH (see Appendix C, Table C-1.3), specifies the path for the output whose value is set with the value of *Output*.

## 5A-40.4 Common Services

The Trip Point Object provides the following Common Services:

**Table 5A-40.5 Trip Point Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0Ehex | Conditional * | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 10hex | n/a | Required | Set_Attribute_Single | Modifies an attribute value. |
| * The Get_Attribute_Single service is REQUIRED if any attributes are implemented. | | | | |
| See Appendix A for description of these services | | | | |

## 5A-40.5 Object–Specific Services

The Trip Point Object provides no Object–Specific services.

## 5A-40.6 Behavior

### 5A-40.6.1 Trip Point Object States

Figure 5A-40.1 Trip Point Object State Transition Diagram

**Table 5A-40.6 DS Instance Behavior State Description**

| State | Description |
|---|---|
| EXECUTING | The Object instance is performing its normal functions. |
| IDLE | The Object instance is not performing its normal function.<br>The Output STATUS attribute is NOT ASSERTED |

Internal requests are coordinated such that the EXECUTING state here will align with the OPERATIONAL state of the Identity object as well as the EXECUTING state of the S-Device Supervisor object in devices where these objects coexist. Transitions to all other states by these objects will cause this object to transition to its IDLE state.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The Activated and Deactivated events correspond to Identity Object events. Where this object is used with an S-Device Supervisor object (see Chapter 0), these events are similarly aligned, but further include alignment with S-Device Supervisor events as specified in Chapter 0.

### 5A-40.6.2 Trip Point Object State Event Matrix

**Table 5A-40.7 State Event Matrix for S-Device Supervisor Object**

| EVENT | STATE | |
|---|---|---|
| | Idle | Executing |
| Power Applied | Default Entry Point | — |
| Identity Object<br>Transition to Operation state<br>(as by an Activated event) | Transition to EXECUTING State | Ignore Event |
| Identity Object<br>Transition from Operation state<br>(as by a Deactivated event) | Ignore Event | Transition to IDLE State |
| S-Device Supervisor Object<br>Transition to Executing State | Transition to EXECUTING State | Ignore Event |
| S-Device Supervisor Object<br>Transition from Executing State | Ignore Event | Transition to IDLE State |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-41 Drive Data Object

**Class Code: Not Assigned**

"This Section is being Reserved for the Drive Data Object."


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-42 File Object

## Class Code: 37 Hex

The number of instances of this object is dependent on the device, and can be obtained by getting the Number_of_Instances class attribute. A device may support creation of new instances, and deletion of these dynamically created instances. Instances built into the product shall not be able to be deleted. Instances of the File Object are divided into the following address ranges to provide for publicly defined and product specific files.

**Table 5A-42.1 File Object Instance ID Ranges**

| Range | Meaning | Quantity |
|---|---|---|
| $1 – C7_{hex}$ | Vendor/Product Specific | 199 |
| $C8_{hex} – FF_{hex}$ | Reserved for CIP (Publicly defined) | 56 |
| $100_{hex} – 4FF_{hex}$ | Vendor/Product Specific | 1024 |
| $500_{hex} – FFFFFFFF_{hex}$ | Reserved for CIP (Publicly defined) | 4,294,966,016 |

## 5A-42.1 Class Attributes

**Table 5A-42.2 File Object Class Attributes**

| Attr ID | Need in implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 2 | These class attributes are optional and are described in Chapter 4 of this Specification. | | | | | |
| 3 | Required | Get | Number_of_Instances | UINT | Number of instances present in this device. | 0 – 65535 |
| 4 thru 7 | These class attributes are optional and are described in Chapter 4 of this volume. | | | | | |
| 32 | Required | Get | Directory | Array of Struct | List of all instance and file names present within this device and the associated instance number. | Number of instance names returned is given in attribute 3, Number_of_Instances. The Instance and File Names are instance attributes 2 and 4, respectively. |
| | | | Instance Number | UINT | | |
| | | | Instance_Name | STRINGI | | |
| | | | File_Name | STRINGI | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-42.2 Instance Attributes

**Table 5A-42.3 File Object Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Required | Get | V | State | USINT | State of the object instance | See 5A-42.6 for details on the object state. Values:<br>0 = Nonexistent<br>1 = File Empty (no file loaded)<br>2 = File Loaded<br>3 = Transfer Upload initiated<br>4 = Transfer Download initiated<br>5 = Transfer Upload in Progress<br>6 = Transfer Download in Progress<br>7 = Storing<br>8 – 255 = Reserved |
| 2 | Required | Get[2] | NV | Instance Name | STRINGI | Name assigned to this instance. | See section 5A-42.2.1.1 |
| 3 | Required | Get[1] | NV[4] | File Format Version | UINT | Format version of the loaded file. | See section 5A-42.2.1.2 |
| 4 | Required | Set[1, 3] | NV[4] | File Name | STRINGI | Name assigned to file loaded in this instance. | See section 5A-42.2.1.3 |
| 5 | Required | Get[1] | NV[4] | File Revision | STRUCT of: | Revision of the file loaded in this instance | See section 5A-42.2.1.4 |
| | | | | Major Revision | USINT | | |
| | | | | Minor Revision | USINT | | |
| 6 | Required | Get[1] | NV[4] | File Size | UDINT | Size of the loaded file. | See section 5A-42.2.1.5 |
| 7 | Required | Get[1] | NV[4] | File Checksum | UINT | Checksum of the loaded file. | See section 5A-42.2.1.6 |
| 8 | Required | Get[2] | NV | Invocation Method | USINT | Method of invoking downloaded file. | 0 = No action required<br>1 = Reset to Identity Object<br>2 = Power cycle on device<br>3 = Start Service request required<br>4 = Application object internal request required<br>5 – 99 = Reserved by CIP<br>100 – 199 = Vendor Specific<br>200 – 254 = Reserved by CIP<br>255 = Not applicable<br>See section 5A-42.2.1.7 |
| 9 | Required | Get | V | File Save Parameters | BYTE | Provides information concerning the non-volatile storage of the file. | See section 5A-42.2.1.8 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**File Object, Class Code: $37_{Hex}$**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 10 | Required | Set[3] | NV | File Access Rule | USINT | Specifies whether the file can be read from and/or written to. | 0 = Read/Write<br>1 = Read Only<br>2 – 255 = Reserved<br>See section 5A-42.2.1.9 |
| 11 | Optional | Get | NV[4] | File Encoding Format | USINT | Defines the encoding format of the data in the file. | 0 = Binary<br>1 = Compressed File(s)<br>2-99 = Reserved by CIP<br>100-199 = Vendor Specific<br>200-254 = Reserved by CIP<br>255 = Unknown<br>See section 5A-42.2.1.10 |

Table Footnotes:

1 This attribute is updated by way of an object specific service.
2 This attribute is assigned by the product in 'factory provided' instances, or during the creation of new object instances.
3 Set for this attribute is optional.
4 The values of these attributes must be saved to non-volatile storage when the content of the file itself is saved to non-volatile storage. Therefore, if the file content has not yet been saved to non-volatile storage (for example, the Save service is required to save the instance, but has not yet been issued), then these attribute values shall not yet be saved to non-volatile storage either.

## 5A-42.2.1 Semantics

### 5A-42.2.1.1 Instance Name – Attribute 2

The Instance Name attribute is used to assign a unique string value to a File Object instance. The name can be assigned by either the product (e.g. product-defined instances) or when a File Object instance is created (e.g. Create service). This attribute cannot be modified via CIP services once the object instance is created. An example of an Instance Name is "Configuration".

Each Instance Name shall include one or more languages in its STRINGI structure. For each language included in a given Instance Name, the string must not be empty (zero length) and must be unique with respect to all other instances' Instance Name strings for that language. Instance Names are not all required to include the same set of languages. Strings for different languages in Instance Names are not required to be unique. The maximum number of languages supported by the device is vendor specific. Devices must support per-language strings of at least 32 characters for dynamically-created instances.

### 5A-42.2.1.2 File Format Version – Attribute 3

The File Format Version value is used to indicate the version of the file format of the loaded file. This is not the version of the file, but rather the version of the format that the loaded file uses. This information, in conjunction with the File Encoding Format attribute when supported, may be used to determine how to interpret the loaded file. A device may also use this information to decide whether to accept a file download operation, based upon support for the particular format version.

If no file is loaded into the instance, then this attribute's value shall be zero (0).

The range and meaning of the File Format Version for vendor-specific files shall be vendor specific. This specification defines the File Format Version values for public file formats in section 5A-42.7, Predefined File Instances.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This attribute shall be updated upon successful completion of loading a file into the File Object instance.

#### 5A-42.2.1.3 File Name – Attribute 4

This text string identifies the name of the file loaded into the instance. For example, an instance with an Instance Name of "Configuration" might be loaded with a file with a File Name of "Product mix 4 run". If no file is loaded into the instance, the File Name shall be empty represented by a STRINGI containing only the Number field with a value of 0x00. If a file is loaded into the instance, the File Name shall include one or more languages in its STRINGI structure. For each language included, the string shall not be empty (zero length). File Names are not all required to include the same set of languages. Duplicate File Names are allowed across instances and languages. The maximum number of languages supported by the device is vendor specific. Devices must support per-language strings of at least 32 characters when this attribute is settable.

This attribute shall be updated upon successful completion of loading a file into the File Object instance.

For instances that support setting this attribute, the File Name may be changed using the Set_Attribute_Single service. However, changing the File Name attribute when no file is loaded into the File Object instance shall not be allowed.

See section 5A-42.5, Error Codes for specific error codes that apply to setting this attribute via the Initiate_Download and Set_Attribute_Single services

#### 5A-42.2.1.4 File Revision – Attribute 5

When files are changed, there is typically a version associated with each revision to keep track of the file's history. This attribute is used to identify which version of a file is loaded into a File Object instance. Both major and minor revision numbers are available. If no file is loaded into the instance, then both Major and Minor revision shall be zero (0). The exact interpretation of the File Revision for vendor-specific files shall be vendor specific. This specification defines the interpretation of File Revision values for public file formats in section 5A-42.7, Predefined File Instances. Some instances may use Major and/or Minor revision values of zero (0).

This attribute shall be updated upon successful completion of loading a file into the File Object instance.

#### 5A-42.2.1.5 File Size – Attribute 6

The File Size attribute is used to indicate the size (in bytes) of the file loaded in the File Object instance. If no file is loaded into the File Object instance, then the File Size shall be zero.

This attribute shall be updated upon successful completion of loading a file into the File Object instance..

#### 5A-42.2.1.6 File Checksum – Attribute 7

The File Checksum attribute contains a 16-bit checksum of the entire file loaded in the File Object instance. If no file is loaded into the instance, then the File Checksum shall be zero (0).

This attribute shall be updated upon successful completion of loading a file into the File Object instance.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The File Checksum attribute is the two's complement of the sixteen-bit sum of all octets in the loaded file. To determine the checksum value, add up every octet in the loaded file, truncate the sum to 16 bits (0x0000 to 0xFFFF), and then subtract the result from 0x10000, again truncating to 16 bits.

Comparing the checksum value between two images (perhaps one loaded into a File Object instance and another held in the client) provides only limited assurance that the files' contents match. If the values differ, then it is known that the contents do not match. However, if the checksum values are the same, then it is only probable that the files match, and a comparison of the full contents would be necessary to determine equality. For example, this checksum algorithm cannot detect transposition of bytes in the file contents or the insertion or deletion of the value zero within the contents.

### 5A-42.2.1.7 Invocation Method – Attribute 8

The Invocation Method attribute indicates what action a device should take upon the successful completion of downloading a file into a File Object instance. The device shall determine the appropriate value for each file instance, including both static instances and dynamic instances. The Invocation Method attribute can have the following values:

**Table 5A-42.4 Invocation Method Values**

| Value | Meaning |
|---|---|
| 0 | No action required.<br>The file is invoked automatically after the new file is loaded. |
| 1 | Reset to Identity Object.<br>The file is invoked only after a Type 0 reset on Identity object instance 1 or a power cycle. |
| 2 | Power cycle on device.<br>The file is invoked only after a physical power cycle. Resets to the Identity object will not cause the file to be invoked. |
| 3 | Start service request required.<br>The file is invoked when the Start service is issued. Resets to the Identity object or power cycle will not cause the file to be invoked. |
| 4 | Application object internal request required.<br>The file is invoked when an internal service is issued by an application object that uses the file. |
| 5 - 99 | Reserved by CIP |
| 100 - 199 | Vendor Specific |
| 200 - 254 | Reserved by CIP |
| 255 | Not applicable.<br>This value shall be used when the device does not itself use the contents of this file. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-42.2.1.8 File Save Parameters – Attribute 9

This attribute provides information about the nonvolatile save of the file. The bit definition is defined below:

**Table 5A-42.5 File Save Parameters Bits**

| Bit(s) | Called | Definition |
|---|---|---|
| 0 | Save Service Required | This bit shall be set if the Save service is required to save the file instance to non-volatile storage after a successful download. This bit shall be clear if this file instance is saved automatically after a successful download. The device determines the value of this bit for each file instance. |
| 1 - 3 | Reserved | Reserved for future use and shall be set to zero. |
| 4 | Save Status | This bit shall be set if the loaded file is equivalent to the copy in non-volatile storage, and clear otherwise. Exception: For Predefined Instances 0xC8 and 0xC9 defined in section 5A-42.7, this bit is always clear. |
| 5 - 7 | Reserved | Reserved for future use and shall be set to zero. |

### 5A-42.2.1.9 File Access Rule – Attribute 10

The File Access Rule attribute indicates if the file can be read and written or only read. Each instance shall define whether or not this attribute is settable.

If this attribute is not settable for the instance, then the instance shall define whether the File Access Rule is fixed at Read/Write (0) or Read Only (1). For example, the Predefined File Instance 0xC8, This Device's EDS File and Icon File, requires that the File Access Rule always be Read Only (1). If this attribute is not settable and Set_Attribute_Single is supported by the instance, then the Set_Attribute_Single service shall return error 0x0E, Attribute Not Settable.

If this attribute is settable for the instance, then the default value shall be Read/Write (0). If no file has been loaded and a request to set this attribute is received, then an Object State Conflict (0x0C) error code shall be returned.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-42.2.1.10 File Encoding Format – Attribute 11

File data may be encoded (e.g. by compressing the data) as defined by this attribute. If this optional attribute is not supported, then there is either no encoding on the data, or the encoding format is simply not reported to the client.

The defined file encoding formats are listed in the following table.

**Table 5A-42.6 File Encoding Format Values**

| Value | Meaning |
|---|---|
| 0 | File should be considered as binary – no character, end of line translation or other interpretation should be applied. |
| 1 | Compressed file or files. Using the ZLIB compression explained in 5A-42.2.1.10.1 |
| 2 - 99 | Reserved by CIP |
| 100-199 | Vendor Specific |
| 200-254 | Reserved by CIP |
| 255 | Unknown – Indicates that no encoding information is available on the file loaded into this instance. This value may be used for files downloaded to an instance that supports this attribute, since the value of this attribute may not be specified by the Initiate_Download service. This value may be used by clients when an instance does not support this attribute.<br>When Unknown is returned, the device and/or client must use an alternate method to determine the actual encoding, such as knowing implicitly what the encoding is for this instance, using a File Name prefix or suffix (extension), or decoding a header within the file content.<br>This value is the default if no file is loaded. |

#### 5A-42.2.1.10.1 Compressed File – ZLIB Encoding (0x01)

The file shall be encoded with the ZLIB compression file format. ZLIB is a free, general purpose, legally unencumbered, loss-less compression library. The specifications are available from the Internet Engineering Task Force (http://www.ietf.org) as:

- RFC 1950, May 1996 (ZLIB Compressed Data Format Specification V.33),
- RFC 1951, May 1996 (DEFLATE Compressed Data Format Specification V. 1.3),
- RFC 1952, May 1996 (GZIP File Format Specification Version 4.3).

A compression and decompression utility and source code are available from the ODVA website (www.ODVA.org). Additional restrictions on the compressed file(s) are:

- Names of files shall be at least one character long, and no more than 32 characters long.
- Characters shall be ASCII characters: 0-9, a-z, A-Z, “.”, “-“, “_” case sensitive.
- Duplicate file names are not allowed.
- Directory structure is not supported.
- File attributes and time/date stamps are not supported.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-42.3 Common Services

**Table 5A-42.7 File Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0x06 | N/A | Optional | Start | Used to enable file |
| 0x07 | N/A | Optional | Stop | Used to disable file |
| 0x08 | Optional | N/A | Create[1] | Creates new File Object instance. |
| 0x09 | N/A | Optional | Delete[1] | Deletes object previously created using the Create service. |
| 0x0E | Required | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 0x10 | N/A | Conditional[2] | Set_Attribute_Single | Writes the contents of the specified attribute. |
| 0x15 | N/A | Optional | Restore | Restores a file from non-volatile storage. |
| 0x16 | N/A | Optional | Save | Saves a loaded file to non-volatile storage. |
| 0x18 | N/A | Conditional[3] | Get_Member | Returns a member element of an array attribute |

Table Footnotes:
1 These services shall be supported together.
2 This service is required when Set is supported on any attributes for this instance. See section 5A-42.5.3 for error codes used by this service.
3 This service shall be supported if Identity attribute 12 is implemented.

See Appendix A for complete description of these services

### 5A-42.3.1 Create Service Description

Creates a new file instance in the device. The Instance_Name parameter is loaded into the Instance Name instance attribute of the newly created object. In addition, the State attribute is set to File Empty indicating that no file is loaded, the File Name attribute is set to be empty and the Revision attribute's Major Revision and Minor Revision members are set to zero. If a Create request is received with an invalid or duplicate Instance_Name as defined by section 5A-42.2.1.1, Instance Name – Attribute 2, then an error is returned indicating Invalid Parameter (Error Code 0x20).

The Instance_Number and Invocation_Method response parameters shall be selected by the File Object using a vendor-specific method. The Instance_Number will be selected from one of the Vendor/Product Specific Instance ID ranges.

**Table 5A-42.8 Create Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Values |
|---|---|---|---|
| Instance_Name | STRINGI | Name applied to this instance. | |

**Table 5A-42.9 Create Response Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Values |
|---|---|---|---|
| Instance_Number | UINT | Instance number allocated | See Table 5A-42.1 for valid Instance IDs |
| Invocation_Method | USINT | Method of invoking loaded file. | See Instance Attribute semantics |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-42.4 Object-specific Services

**Table 5A-42.10 File Object Object-specific Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0x4B | N/A | Required | Initiate_Upload | Used to start a file upload |
| 0x4C | N/A | Conditional [1] | Initiate_Download | Used to start a file download |
| 0x4D | N/A | Optional | Initiate_Partial_Read | Used to start a partial read of a file |
| 0x4E | N/A | Conditional [2] | Intitiate_Parial_Write | Used to start a partial write of a file |
| 0x4F | N/A | Required | Upload_Transfer | Performs a file transfer upload |
| 0x50 | N/A | Conditional [1] | Download_Transfer | Performs a file transfer download |
| 0x51 | N/A | Conditional [1] | Clear File | Clears a loaded file |

Table Footnotes:

1. These services are required when the File Access Rule attribute on this instance allows the Read/Write value. Otherwise, these services shall not be supported.
2. This service is optional when the File Access Rule attribute on this instance allows the Read/Write value. Otherwise this service shall not be supported.

### 5A-42.4.1 Initiate_Upload Service Description

Initiates a file upload. The client indicates the maximum transfer size it can handle. The server indicates total file size and actual transfer size. The device updates the State attribute to 'Transfer Upload Initiated' if successful.

**Table 5A-42.11 Initiate_Upload Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Values |
|---|---|---|---|
| Maximum Transfer Size | USINT | Maximum number of bytes the client can accept in each transfer service. | |

**Table 5A-42.12 Initiate_Upload Response Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Values |
|---|---|---|---|
| File Size | UDINT | Size of file to upload, in bytes. This value may be the maximum number of bytes, which can be transferred from this instance, and not the actual file size. | |
| Transfer Size | USINT | Maximum number of bytes the device will deliver in the File_Data parameter of each Upload Transfer response. This value shall be equal to or less than the Maximum Transfer Size specified in the request. The server may send less than this amount. | |

### 5A-42.4.2 Initiate_Download Service Description

Initiates a file download. The client indicates total file size, file format version, file revision, and file name. The server returns the transfer size, number of bytes for each 'save to non-volatile memory' and the time required to do the save operation. The device updates the State attribute to 'Transfer Download Initiated' on success. If an Initiate_Download request is received with an invalid File Name parameter, the request shall be rejected. See sections 5A-42.2.1.3 and 5A-42.5.1 for additional details on validating the File Name parameter and the appropriate error codes.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-42.13 Initiate_Download Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Values |
|---|---|---|---|
| File Size | UDINT | Size of file to download, in bytes. This value may be the maximum number of bytes, which can be transferred from this instance, and not the actual file size. | |
| File Format Version | UINT | File Format Version of the file to be downloaded. See section 5A-42.2.1.2 for details. The server shall verify that this file format is supported by this file object instance. | |
| File Revision | USINT<br>USINT | Major Revision<br>Minor Revision<br>These values are used to update the File Revision instance attribute after successful completion of the download. | |
| File Name | STRINGI | User assigned name of file to download. This value is used to update the File Name instance attribute after successful completion of the download. | |

**Table 5A-42.14 Initiate_Download Response Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Values |
|---|---|---|---|
| Incremental Burn | UDINT | Number of bytes before a save to non-volatile memory occurs. The save operation may cause the device to delay responding to the transfer which causes a non-volatile save to occur. | |
| Incremental Burn Time | UINT | Number of seconds required to perform the save to non-volatile memory. | 0 – 65535 seconds |
| Transfer Size | USINT | Maximum number of bytes the device will accept the File_Data parameter of each Download Transfer request. The client may send less than this amount. | |

### 5A-42.4.3 Initiate_Partial_Read Service Description

Initiates a partial file read. The client indicates the file offset, number of bytes to read, and maximum transfer size it can handle. The server shall verify that the entire data range exists in the file. The server returns the read size (which shall match the requested read size) and actual transfer size. The device updates the State attribute to 'Transfer Upload Initiated' if successful.

**Table 5A-42.15 Initiate_Partial_Read Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Values |
|---|---|---|---|
| File Offset | UDINT | Byte offset into the file where the read begins. | |
| Read Size | UDINT | Number of bytes to read, starting at File Offset. | |
| Maximum Transfer Size | USINT | Maximum number of bytes the client can accept in each transfer service. | |

The Initiate_Partial_Read Response Parameters are identical to the Initiate_Upload Response parameters.

### 5A-42.4.4 Initiate_Partial_Write Service Description

Initiates a partial file write. The client indicates file offset, write size, file format version, and file revision. The server shall verify that the entire data range exists in the file. The server returns the transfer size, number of bytes for each 'save to non-volatile memory', and the time required to do the save operation. The device updates the State attribute to 'Transfer Download Initiated'.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-42.16 Initiate_Partial_Write Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Values |
|---|---|---|---|
| File Offset | UDINT | Byte offset into the file where the write begins. | |
| Write Size | UDINT | Number of bytes to write, starting at File Offset | |
| File Format Version | UINT | File Format Version of the file to be written to. See section 5A-42.2.1.2 for details. The server shall verify that this file format matches or is compatible with the File Format Version of the file loaded in this instance. | |
| File Revision | USINT<br>USINT | Major Revision<br>Minor Revision<br>These values are used to update the File Revision instance attribute after successful completion of the download. | |

The Initiate_Partial_Write Response Parameters are identical to the Initiate_Download Response parameters.

### 5A-42.4.5 Upload_Transfer Service Description

Performs a file upload. The client and server both verify that an 8-bit value (the Transfer Number), which is sent by the client and returned by the server, is correct (starts at zero and increments by one on each transfer). The State attribute is updated at the first ('Transfer Upload in Progress') and last ('File Loaded' or 'Empty, No File Loaded') transfers. The last transfer packet contains a 16-bit checksum appended to the end of the file data, which is matched by the client to what it calculated. The checksum is not included in the file size.

The server shall send the next packet if the transfer number is one more than the last (or zero and this is either the first packet or a rolled over count), resend the last packet if the transfer number is the same as the last request, or send an error response (error code $20_{hex}$, Invalid Parameter and Additional Error Code $06_{hex}$) if the transfer number is neither of the above values.

**Table 5A-42.17 Upload_Transfer Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Values |
|---|---|---|---|
| Transfer Number | USINT | This 8 bit value starts at zero and is incremented on each additional transfer. The value 'rolls over' to zero after 255. The server uses this value to verify the correct packet is received. | 0-255=Transfer number |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-42.18 Upload_Transfer Response Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Values |
|---|---|---|---|
| Transfer Number | USINT | The transfer number is an 8 bit value echoing back the transfer number received. | 0 – 255 = Transfer number |
| Transfer Packet Type | USINT | The transfer packet type provides 'first', 'middle', and 'last' transfer information, as well as the ability to terminate the transfer. The client uses this value to verify integrity of the transfer, and to determine the completion of the transfer. | 0 = First transfer packet<br>1 = Middle transfer packet<br>2 = Last transfer packet<br>3 = Abort transfer<br>4 = First and last packet<br>5 – 255 = Reserved |
| File_Data | Array of BYTE | File transfer data. | |
| Transfer Checksum | INT | Checksum of all transferred file data, only on last transfer. | |

### 5A-42.4.6 Download_Transfer Service Description

Performs a file download. The client and server both verify the Transfer Number is correct (starts at zero and increments by one on each transfer). Upon receipt of the first transfer the State attribute is updated to 'Transfer Download in Progress'. The last transfer packet contains a 16 bit checksum appended to the end of the file data which is matched by the client to what it calculated. The checksum is not included in the file size. Upon completion, the server shall verify the checksum and then update the State attribute ('File Loaded' or 'File Empty'). If the operation was successful, the File Name, File Revision, File Size, and File Checksum attributes are updated. If not, the object transitions to the File Empty state.

When this service is received, the server shall either accept the packet if the transfer number is one more than the last (or zero and this is either the first packet or a rolled over count), discard the packet if the transfer number is the same as the last request, or send an error response (error code $20_{hex}$, Invalid Parameter and Additional Error Code $06_{hex}$) if the transfer number is neither of the above values. All error conditions and codes for this service are described in the table below. This object defines one Object Class specific error code.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-42.19 Download_Transfer Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Values |
|---|---|---|---|
| Transfer Number | USINT | The transfer number is an 8 bit value which starts at zero and is incremented on each additional transfer. The value 'rolls over' to zero after 255. The server uses this value to verify integrity of the transfer. | 0 – 255 = Transfer number |
| Transfer Packet Type | USINT | The transfer packet type provides 'first', 'middle', and 'last' transfer information, as well as the ability to terminate the transfer. The server uses this value to verify integrity of the transfer, and to determine the completion of the transfer. | 0 = First transfer packet<br>1 = Middle transfer packet<br>2 = Last transfer packet<br>3 = Abort transfer<br>4 = First and last transfer<br>5 – 255 = Reserved |
| File_Data | Array of USINT | File transfer data. | |
| Transfer Checksum | UINT | Checksum of all transferred file data, only on last transfer (Transfer Packet Type = Last). | |

**Table 5A-42.20 Download_Transfer Response Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Values |
|---|---|---|---|
| Transfer Number | USINT | Transfer number echoed back to client. The client uses this value to verify integrity of the transfer. | 0-255=Transfer number |

### 5A-42.4.7 Clear_File Service Description

This object-specific instance service clears (deletes contents of) a loaded file from both volatile and nonvolatile memory. There are no parameters defined. In addition to the deletion of the file data, the State is set to File Empty and the following attributes of the instance are cleared:

- File_Name
- File Revision
- File_Size
- File Checksum
- File Save Parameters


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-42.5 Error Codes

### 5A-42.5.1 Error codes for Initiate Services

The following table defines error codes that are used by the Initiate_Upload, Initiate_Download, Initiate_Partial_Read and Initiate_Partial_Write services.

**Table 5A-42.21 Error Codes for Initiate Services**

| Error Condition | General Status | Extended Status |
|---|---|---|
| OBSOLETE - File Size too large | 0x20 | 0x0000 |
| OBSOLETE - File Format Version not Compatible | 0x20 | 0x0001 |
| File Size too large | 0x20 | 0x0004 |
| File Format Version not Compatible | 0x20 | 0x0005 |
| Fail on transfer – zero size | 0x20 | 0x0008 |
| File Name empty[1] | 0x20 | 0x000A |
| File name too long[1] | 0x15 | 0x0001 |
| Too many languages in File Name[1] | 0x15 | 0x0002 |
| File empty[2] | 0x0C | |
| File Offset out of range[3] | 0x20 | 0x0002 |
| Read/Write Size goes beyond end of file[3] | 0x20 | 0x0003 |

Table Footnotes
1. These errors only apply to the Initiate_Download service.
2. This error only applies to the Initate_Upload, Initiate_Partial_Read and Intiate_Partial_Write services.
3. These errors only apply to the Initiate_Partial_Read and Intiate_Partial_Write services.

### 5A-42.5.2 Error codes for Transfer Services

The following table defines error codes that are used by the Download_Transfer and Upload_Transfer services.

**Table 5A-42.22 Error Codes for Transfer Services**

| Error Condition | General Status | Extended Status |
|---|---|---|
| OBSOLETE - Fail on transfer – out of sequence | 0x20 | 0x0000 |
| OBSOLETE - Fail on transfer – other | 0x20 | 0x0001 |
| Fail on transfer – out of sequence | 0x20 | 0x0006 |
| Fail on transfer – other | 0x20 | 0x0007 |
| Fail on transfer – zero size | 0x20 | 0x0008 |
| OBSOLETE – Duplicate File Name | 0x20 | 0x0009 |
| Fail on checksum | 0xD0 | |
| Fail on save to non-volatile memory | 0x19 | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-42.5.3 Error Codes for the Set_Attribute_Single Service

The following table defines error codes that are used by the Set_Attribute_Single service.

**Table 5A-42.23 Error Codes for the Set_Attribute_Single Service**

| Error Condition | General Status | Extended Status |
|---|---|---|
| Service not supported[1] | 0x08 | |
| Attribute not settable[2] | 0x0E | |
| File empty[3] | 0x0C | |
| File Access Rule is read only[4] | 0x0C | |
| File Name empty[5] | 0x20 | 0x000A |
| File Name too long[5] | 0x15 | 0x0001 |
| Too many languages in File Name[5] | 0x15 | 0x0002 |

Table Footnotes:

1 This error shall only be returned when this instance does not support Set_Attribute_Single for any attributes.

2 This error shall be returned for any supported attribute that is not settable for this file instance. Notice that this error does not apply when the File Access Rule attribute is settable and currently set to Read Only (1) and the File Name attribute is being set.

3 This error shall be returned for any settable attributes when the file instance is in the File Empty state.

4 This error shall be returned when the File Access Rule attribute is settable and currently set to Read Only (1) and the File Name attribute is being set.

5 These errors only apply when the Set_Attribute_Single is requested for the File Name attribute.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-42.6 Behavior

The State Transition Diagram and State Event Matrix table below illustrate the behavior of the File Object.

**Figure 5A-42.1 State Transition Diagram for File Object Instance**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-42.24 File Object Instance State Event Matrix**

| Event | File Object State | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **Non-Existent** | **File Empty** | **Transfer Download Initiated** | **Transfer Download in Progress** | **Storing** | **File Loaded** | **Transfer Upload Initiated** | **Transfer Upload in Progress** |
| Power Up | *If* object exists then *If* file loaded transition to File Loaded *Else* transition to File Empty | Not applicable | Not applicable | Not applicable | Not applicable | Not applicable | Not applicable | Not applicable |
| Create | *If* Instance Name unique transition to File Empty *Else* Return error: Invalid Parameter | Not applicable | Not applicable | Not applicable | Not applicable | Not applicable | Not applicable | Not applicable |
| Initiate Download [5] | Return error: Object Does Not Exist | Process service data[1], transition to Transfer Download Initiated | Clear previous service data[2], process new service data[1] | Clear all file data and attributes[3], process new service data[1], transition to Transfer Download Initiated | Return error: Object State Conflict | Clear all file data and attributes[3], process service data[1], transition to Transfer Download Initiated | Return error: Object State Conflict | Return error: Object State Conflict |
| Initiate Partial Write[5] | Return error: Object Does Not Exist | Return error: Object State Conflict | Process new service data[6] | Process new service data[6], transition to Transfer Download Initiated | Return error: Object State Conflict | Process service data[6], transition to Transfer Download Initiated | Return error: Object State Conflict | Return error: Object State Conflict |
| Download Transfer, First Packet [5] | Return error: Object Does Not Exist | Return error: Object State Conflict | Process first block of data, transition to Transfer Download in Progress | Return error: Object State Conflict, clear all file data and attributes[3], transition to File Empty | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict |
| Download Transfer, Middle Packet [5] | Return error: Object Does Not Exist | Return error: Object State Conflict | Return error: Object State Conflict, transition to File Empty | Process data | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## File Object, Class Code: $37_{Hex}$

| Event | File Object State | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **Non-Existent** | **File Empty** | **Transfer Download Initiated** | **Transfer Download in Progress** | **Storing** | **File Loaded** | **Transfer Upload Initiated** | **Transfer Upload in Progress** |
| Download Transfer, Last Packet [5] | Not applicable | Return error: Object State Conflict | Return error: Object State Conflict, transition to File Empty | Process data, checksum. *If* not valid return error: Invalid Parameter and transition to File Empty *Else* update attribute values and *If* auto-save transition to Storing *Else* transition to File Loaded | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict |
| Nonvolatile storage complete | Not applicable | Ignore Event | Ignore Event | Ignore Event | Transition to File Loaded | Ignore Event | Ignore Event | Ignore Event |
| Initiate Upload/ Initiate Partial Read | Return error: Object Does Not Exist | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Process service data[4], transition to Transfer Upload Initiated | Process new service data[4] | Process new service data[4], transition to Transfer Upload Initiated |
| Upload Transfer, First Packet | Return error: Object Does Not Exist | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Retrieve first block of data, transition to Transfer Upload in Progress | Return error: Object State Conflict, transition to File Loaded |
| Upload Transfer, Middle Packet | Return error: Object Does Not Exist | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict, transition to File Loaded | Retrieve data |
| Upload Transfer, Last Packet | Return error: Object Does Not Exist | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict, transition to File Loaded | Retrieve data and checksum, transition to File Loaded |
| Clear [5] | Return error: Object Does Not Exist | Return success | Clear all file data, Transition to File Empty | Clear all file data, Transition to File Empty | Clear all file data, Transition to File Empty | Clear all file data, Transition to File Empty | Clear all file data, Transition to File Empty | Clear all file data, Transition to File Empty |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## File Object, Class Code: $37_{Hex}$

| Event | File Object State | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **Non-Existent** | **File Empty** | **Transfer Download Initiated** | **Transfer Download in Progress** | **Storing** | **File Loaded** | **Transfer Upload Initiated** | **Transfer Upload in Progress** |
| Transfer Failure/Abort Transfer | Not Applicable | Not applicable | Transition to File Empty | Clear all file data and attributes[3], transition to File Empty | Not applicable | Not applicable | Transition to File Loaded | Transition to File Loaded |
| Delete | Return error: Object Does Not Exist | Transition to Non-Existent | Clear all file data and attributes[3]. Transition to Non-Existent | Clear all file data and attributes[3]. Transition to Non-Existent | Clear all file data and attributes[3]. Transition to Non-Existent | Clear all file data and attributes[3]. Transition to Non-Existent | Clear all file data and attributes[3]. Transition to Non-Existent | Clear all file data and attributes[3]. Transition to Non-Existent |
| Save | Return error: Object Does Not Exist | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Save file data and attributes to nonvolatile storage. | Return error: Object State Conflict | Return error: Object State Conflict |
| Restore | Return error: Object Does Not Exist | Restore file data and attributes from nonvolatile storage, transition to File Loaded. | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Clear all file data and attributes[3]. Restore new file data and attributes from nonvolatile storage. | Return error: Object State Conflict | Return error: Object State Conflict |
| Download Transfer, First-and-last Packet [5] | Return error: Object Does Not Exist | Return error: Object State Conflict | Same as Download Transfer First Last Packet in the Download in Progress state | Return error: Invalid Parameter, transition to File Empty | Same as Download Transfer Last Packet | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Event | File Object State | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **Non-Existent** | **File Empty** | **Transfer Download Initiated** | **Transfer Download in Progress** | **Storing** | **File Loaded** | **Transfer Upload Initiated** | **Transfer Upload in Progress** |
| Upload Transfer, First-and-last Packet | Return error: Object Does Not Exist | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Return error: Object State Conflict | Same as Upload Transfer Last Packet in the Transfer Upload in Progress state | Return error: Object State Conflict, transition to File Loaded |

Table Footnotes:
1 Processing the Initiate Download service data requires temporary storage of the file name, file size, format version, and file revision and an initialization of the internal transfer variables (file location, transfer number, etc.).
2 Clearing the previous Initiate Download service data requires deletion of the parameters stored temporarily from the previous Initiate Download service request.
3 Clearing all file data and attributes requires the deletion of all file data and settable attribute values (file name, file size, etc.) from volatile memory (or from non-volatile memory if data storage is only in non-volatile storage). If the device has only non-volatile storage for this data than the file is lost during this process. If the device has separate volatile and non-volatile storage for the file, it may be recovered by a power cycle or the Restore service.
4 Processing the Initiate Upload and Initiate Partial Read service data requires resolution/retention of the maximum transfer size and an initialization of the internal transfer variables (file location, transfer number, etc.).
5 If this service is supported and the File Access Rule attribute value is Read Only, then this service shall be rejected with the error response Object State Conflict (0x0C).
6 Processing the Initiate Partial Write requires initialization of the internal transfer variables (file location, transfer number, etc.).

## 5A-42.7 Predefined File Instances

Predefined instances of the File class allow for special purpose files to exist on any product, regardless of vendor or device type. Each predefined file defines non-volatile attribute values, services supported and any other type of behavior. The following table lists these publicly defined files.

**Table 5A-42.25 Table of Predefined File Instances**

| Instance Number | File Description | Section |
|---|---|---|
| 0xC8 | This Device's EDS File and Icon File | 5A-42.7.1 |
| 0xC9 | Collection of related EDS and Icon files | 5A-42.7.2 |
| 0xCA – 0xFF | Reserved by CIP | |

### 5A-42.7.1 This Device's EDS File and Icon File – (0xC8)

This instance shall contain the EDS file for the device (the device containing this instance), and optionally the Icon file for the device. If this is a device that uses Modular EDSs, then this is the EDS file and Icon for the adapter. One of the following shall be present in this instance:

- Uncompressed EDS file for this device (File Encoding Format 0)
- Compressed EDS file for this device (File Encoding Format 1)
- Compressed EDS and Icon files for this device (File Encoding Format 1)

The following tables describe the services supported and attribute values for this instance.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-42.26 Table of Required Attributes for File Instance 0xC8**

| Attribute ID | Access Allowed | Attribute Name | Attribute Value |
|---|---|---|---|
| 1 | Get | State | See section 5A-42.2 |
| 2 | Get | Instance Name | "EDS and Icon Files"[1] |
| 3 | Get | File Format Version | 1 |
| 4 | Get | File Name | "EDS.gz"[1] if compressed<br>"EDS.txt"[1] if not compressed |
| 5 | Get | File Revision | Shall be the same as the EDS Revision field that is within the EDS file. |
| 6 | Get | File Size | See section 5A-42.2.1.5 |
| 7 | Get | File Checksum | See section 5A-42.2.1.6 |
| 8 | Get | Invocation Method | 255 |
| 9 | Get | File Save Parameters | 0 |
| 10 | Get | File Access Rule | 1 |
| 11 | Get | File Encoding Format | 0 or 1 |

Table Footnotes:

1 This is the English ('eng') language string that shall be included in the STRINGI structure. Other language strings may be included but are not defined in this specification. The English language string shall use one of the STRING, SHORT_STRING, or STRING2 structures and shall use CharSet 4 (ISO-8859-1) for STRING and SHORT_STRING types and CharSet 1000 (ISO-10646-UCS-2) for the STRING2 type.

**Table 5A-42.27 Table of Service Codes for File Instance 0xC8**

| Service Code | Service Name | Need In Implementation |
|---|---|---|
| 0x0E | Get_Attribute_Single | Required |
| 0x4B | Initiate Upload | Required |
| 0x4D | Initiate Partial Read | Optional |
| 0x4F | Upload Transfer | Required |

### 5A-42.7.2 Collection of Related EDS and Icon Files – (0xC9)

This instance shall contain a collection of 1 or more EDS and Icon files related to this device, as well as the Icon file for this device if the EDS file (contained in instance 0xC8) is stored uncompressed. This instance could be applied in support of modular systems, although it is not restricted to that use. This instance shall not contain any files that are included in instance 0xC8 (defined in section 5A-42.7.1).

The following tables describe the services supported and attribute values for this instance.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-42.28 Table of Required Attributes for File Instance 0xC9**

| Attribute ID | Access Allowed | Attribute Name | Attribute Value |
|---|---|---|---|
| 1 | Get | State | See section 5A-42.2 |
| 2 | Get | Instance Name | "Related EDS and Icon Files"[1] |
| 3 | Get | File Format Version | 1 |
| 4 | Get | File Name | "EDSCollection.gz" [1] |
| 5 | Get | File Revision | This is a file revision for the entire collection and not necessarily the revision of any file within the collection. |
| 6 | Get | File Size | See section 5A-42.2.1.5 |
| 7 | Get | File Checksum | See section 5A-42.2.1.6 |
| 8 | Get | Invocation Method | 255 |
| 9 | Get | File Save Parameters | 0 |
| 10 | Get | File Access Rule | 1 |
| 11 | Get | File Encoding Format | 1 |

Table Footnotes:

1 This is the English ('eng') language string that shall be included in the STRINGI structure. Other language strings may be included but are not defined in this specification. The English language string shall use one of the STRING, SHORT_STRING, or STRING2 structures and shall use CharSet 4 (ISO-8859-1) for STRING and SHORT_STRING types and CharSet 1000 (ISO-10646-UCS-2) for the STRING2 type.

**Table 5A-42.29 Table of Service Codes for File Instance 0xC9**

| Service Code | Service Name | Need In Implementation |
|---|---|---|
| 0x0E | Get_Attribute_Single | Required |
| 0x4B | Initiate Upload | Required |
| 0x4D | Initiate Partial Read | Optional |
| 0x4F | Upload Transfer | Required |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-43 S-Partial Pressure Object

**Class Code: 38 Hex**

An S-Partial Pressure Object provides the means for scanning the partial pressures of a selected Atomic Mass Units (AMU) or gas species. Instances of this object are configured for: AMU, AMU range, gas species or Total Pressure. Class Level services are provided for: requesting a list of configured instances; requesting a list of enabled partial pressures above their respective report thresholds; requesting a list of all enabled partial pressures, irrespective of their report threshold; and, enabling/disabling a group of instances. Also, instance create, delete services are supported.

This object is a member of the Hierarchy of Semiconductor Equipment Devices. As such, its behavior is managed by the S-Device Supervisor Object. See the object definition in Chapter 0.

The S-Partial Pressure object makes use of a list of Standard Gas Type Numbers. This list is described in publication:

SEMI E52-95 “Practice for Referencing Gases Used in Digital Mass Flow Controllers”, Semiconductor Equipment and Materials International (SEMI), Mountain View, CA 94043-4080.

Note: It is implied that the reference above is to the latest revision as specified by SEMI.

## 5A-43.1 Class Attributes

**Table 5A-43.1 S-Partial Pressure Object Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 - 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | | |
| 32 | Required | Set | V | Filament Control | BYTE | Enables a filament to turn on if not interlocked | Bit mapped byte<br>see Semantics section |
| 33 | Required | Get | V | Filament Status | BYTE | Indicates that a filament is on or off and if a filament fault condition exists | Bit mapped byte<br>see Semantics section |
| 34 | Required | Get | V | Reading Invalid | BOOL | A device will indicate Reading Invalid if it determines that one or more of its operational parameters are beyond factory set limits. | 0 = Valid<br>1 = Invalid |
| 35 | Optional | Set | NV | Emission Current | REAL | Filament emission current | AMPS |
| 36 | Optional | Set | V | Multiplier On | BOOL | Turns on the electron multiplier | 0 = Off<br>1 = On |
| 37 | Optional | Set | NV | Multiplier Voltage | REAL | Sets the multiplier field excitation voltage | VOLTS |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**S-Partial Pressure Object, Class Code: $38_{Hex}$**

| Attribute ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 38 | Optional | Set | NV | Samples | USINT | specifies the number of samples configured for each enabled instance | 0 = manufacturer default [default] |
| 39 | Optional | Set | NV | Scan Count | UINT | increments by one upon completion of each Measurement Scan ** | 0 - 65535<br>(value after 65535 is 0) |
| 97 - 98 | Reserved by CIP for Future Use | | | | | | |
| 99 | Conditional * | Get | NV | Subclass | UINT | Identifies a subset of additional attributes, services and behaviors. The subclasses for this object are specified at the end of this object specification section | 0 = No Subclass<br>1 – 65535 = Reserved |

* If the value of this attribute is 0, then its support is OPTIONAL, otherwise support is REQUIRED

** A Measurement Scan is considered to be one complete cycle of measurements performed by the Device.

### 5A-43.1.1 Class Level Subclasses

Each class level subclass defines a unique meaning for an overlapping range of class attribute IDs and/or class service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given class is identified by the value of its Subclass class attribute. The subclasses for this object are specified at the end of this object specification section.

### 5A-43.1.2 Semantics

#### 5A-43.1.2.1 Filament Control

A bit that is set to 1 (one) in the *Control* byte enables the corresponding filament to turn on. Once enables, a filament will turn on if it is able. Various external or internal interlocks may prevent the filament from turning on.

**Table 5A-43.2 Filament Control Attribute**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Reserved | Reserved | Reserved | Reserved | Filament 4 Enable | Filament 3 Enable | Filament 2 Enable | Filament 1 Enable |

An attempt to set a *Filament Enable* bit corresponding to a filament not supported by the device will return an *Invalid Attribute Value* error, as will an attempt to set a filament state not supported by the device (e.g., multiple filaments enabled).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5A-43.1.2.2 Filament Status

If a filament is on, the corresponding *Filament On* bit in the *Status* byte is set to 1 (one). If a filament has been determined by the device to be in a fault state (such as a burned out or broken filament), the corresponding *Filament Fault* bit on the *Status* byte is set to 1 (one).

**Table 5A-43.3 Filament Status Attribute**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Filament 4 Fault | Filament 3 Fault | Filament 2 Fault | Filament 1 Fault | Filament 4 On | Filament 3 On | Filament 2 On | Filament 1 On |

### 5A-43.2 Instance Attributes

Certain minimal implementations may support some optional “Set” attributes as “Get” only and still be compliant with this object specification. All required attributes must be supported as specified.

**Table 5A-43.4 S-Partial Pressure Object Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Optional | Get | V | Number of Attributes | USINT | Number of supported attributes | The number of attributes supported by this object instance |
| 2 | Optional | Get | V | Attribute List | ARRAY OF USINT | List of supported attributes | The list of attributes supported by this object instance |
| 3 | Required | Set | NV | Instance Type | USINT | Specifies the measurement type for the instance | 0 = AMU [default]<br>1 = AMU Range<br>2 = Gas Species<br>3-50 = reserved<br>51-99= Device Specific<br>100-255 = Vendor Specific |
| 4 | Conditional * | Set | NV | AMU | REAL | Atomic Mass Unit (or Starting AMU for analog) | [default] = 0<br>see Semantics section |
| 5 | Conditional * | Set | NV | Ending AMU | REAL | Ending AMU for an AMU Range type instance | see Semantics section |
| 6 | Conditional * | Set | NV | Gas Standard Number | UINT | Gas Species Type Number | [default] = 0 (no gas type specified)<br>see Semantics section |
| 7 | Required | Get | V | Status | BYTE | Alarm and Warning State of this object instance | see Semantics section |
| 8 | Optional | Set | NV | Alarm High Enable | BOOL | Enables the setting of the Alarm High Status Bits | 0 = disable [default]<br>1 = enable |
| 9 | Optional | Set | NV | Alarm Low Enable | BOOL | Enables the setting of the Alarm Low Status Bits | 0 = disable [default]<br>1 = enable |
| 10 | Optional | Set | NV | Warning High Enable | BOOL | Enables the setting of the Warning High Status Bits | 0 = disable [default]<br>1 = enable |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## S-Partial Pressure Object, Class Code: $38_{Hex}$

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 11 | Optional | Set | NV | Warning Low Enable | BOOL | Enables the setting of the Warning Low Status Bits | 0 = disable [default]<br>1 = enable |
| 12 | Required | Set | NV | Instance Enable | USINT | Enables the measurement of this AMU | 0 = disable [default]<br>1 = normal scan rate<br>2 = double scan rate<br>3-50 = reserved<br>51-99= Device Specific<br>100-255 = Vendor Specific<br>see Semantics section |
| 13 | Required | Get | V | Partial Pressure | REAL | The measured Partial Pressure of the specified AMU | The measured value of partial pressure for this AMU.<br>Units are specified by attribute Pressure Units. |
| 14 | Optional | Set | NV | Pressure Units | ENGUNITS | Determines the units context of the Partial Pressure attribute | see Appendix D for a list of values.<br>[default] = Torr<br>If attribute is not supported, [default] is implied. |
| 15 | Optional | Set | NV | Dwell Time | UINT | Determines the acquisition time for this AMU | Time in Milliseconds<br>0 = fastest possible [default] |
| 16 | Optional | Set | NV | Electron Energy | REAL | Indicates the Electron Impact Ionization Voltage | Units = eV<br>[default] = Factory Configured |
| 17 | Optional | Set | NV | Report Threshold | REAL | Trip point for reporting in the Get_Pressures service response | The level above which the Partial Pressure will be reported.<br>see Class Level Services<br>[default] = 0 |
| 18 | Optional | Set | NV | Alarm Trip Point High | REAL | Determines the Value above which an Alarm Condition will occur | see Semantics section<br>[default] = Maximum value for its data type. |
| 19 | Optional | Set | NV | Alarm Trip Point Low | REAL | Determines the Value below which an Alarm Condition will occur | see Semantics section<br>[default] = Minimum value for its data type. |
| 20 | Optional | Set | NV | Alarm Hysteresis | REAL | Determines the amount by which the Value must recover to clear an Alarm Condition | see Semantics section<br>[default] = 0 |
| 21 | Optional | Set | NV | Warning Trip Point High | REAL | Determines the Value above which a Warning Condition will occur | see Semantics section<br>[default] = Maximum value for its data type. |
| 22 | Optional | Set | NV | Warning Trip Point Low | REAL | Determines the Value below which a Warning Condition will occur | see Semantics section<br>[default] = Minimum value for its data type. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**S-Partial Pressure Object, Class Code: $38_{Hex}$**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 23 | Optional | Set | NV | Warning Hysteresis | REAL | Determines the amount by which the Value must recover to clear a Warning Condition | see Semantics section<br>[default] = 0 |
| 24 | Optional | Set | NV | Group ID | USINT | Identifies this instance as belonging to the specified group | 0 - 255 groups |
| 97 - 98 | Reserved by CIP for Future Use | | | | | | |
| 99 | Conditional ** | Get | NV | Subclass | UINT | Identifies a subset of additional attributes, services and behaviors. The subclasses for this object are specified at the end of this object specification section. | 0 = No Subclass<br>1 - 65535 = Reserved |

* Which of these parameterization attributes are supported depends upon the allowable values for the Instance Type attribute.
** If the value of Subclass is 00, which identifies "no subclass", then this attribute is OPTIONAL in implementation, otherwise, this attribute is REQUIRED.

### 5A-43.2.1 Instance Level Subclasses

Each instance level subclass defines a unique meaning for an overlapping range of instance attribute IDs and/or instance service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given instance is identified by the value of its Subclass instance attribute. The subclasses for this object are specified at the end of this object specification section.

### 5A-43.2.2 Semantics

#### 5A-43.2.2.1 Instance Type

The vendor must specify which values are supported. An attempt to set this attribute to an unsupported value will return an Invalid Attribute Value error.

#### 5A-43.2.2.2 AMU

The *AMU* attribute is used to configure the instance for a particular Atomic Mass Unit. For basic AMU type configurations, this specifies the point at which the partial pressure is determined. For AMU Range type configurations, this specifies the point at which the range of AMU begins. A value of zero (0) configures the object instance for Total Pressure. Total Pressure is defined as the sum of all enabled S-Partial Pressure object instances. For Gas Species type configurations, this value is ignored.

#### 5A-43.2.2.3 Ending AMU

For basic AMU type configurations, this value is ignored. For AMU Range type configurations, this specifies the point at which the range of AMU ends. For Gas Species type configurations, this value is ignored.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-43.2.2.4 Gas Standard Number

The Gas Standard Number attribute is used to configure the instance for a particular gas species. For basic AMU, and AMU Range, type configurations, this value is ignored. For Gas Species type configurations, this specifies the Gas Standard Number for the particular gas whose partial pressure is being measured and reported.

The actual coding of the values are described in the following publication:

- See introduction section (above) for reference to the SEMI publication:
- "Practice for Referencing Gases Used in Digital Mass Flow Controllers".

### 5A-43.2.2.5 Status

Status is a bit mapped byte, which indicates the Alarm and Warning Exception status of the object instance. The following definition applies:

**Table 5A-43.5 Status Attribute**

| Bit | Definition |
|---|---|
| 0 | Alarm High Exception: 0 = cleared; 1 = set |
| 1 | Alarm Low Exception: 0 = cleared; 1 = set |
| 2 | Warning High Exception: 0 = cleared; 1 = set |
| 3 | Warning Low Exception: 0 = cleared; 1 = set |
| 4 | Reserved |
| 5 | Reserved |
| 6 | Reserved |
| 7 | Reserved |

### 5A-43.2.2.6 Instance Enable

The *Instance Enable* attribute is used to turn on/off the object instance. Although an instance may be completely configured, if *disabled*, the device will skip the partial pressure measurement in its scan routine.

Due to the time dependent nature of measuring a partial pressure, a feature is included that allows for a double scan rate. This mode of operation causes the device to measure this AMU twice as often as it normally would. This yields a faster update rate for this AMU's partial pressure measurement.

### 5A-43.2.2.7 Dwell Time and Electron Energy

In the event that one of these values is modified while the device is actively acquiring a corresponding reading, the measurement in process shall proceed as originally configured and the new values shall take effect on subsequent readings.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5A-43.2.2.8 Trip Points and Hysteresis

A Trip Point High value is that value above which the *Value* attribute will cause an Alarm or Warning exception condition.

A Trip Point Low value is that value below which the *Value* attribute will cause an Alarm or Warning exception condition.

A Hysteresis value specifies the amount by which the *Value* attribute must transition in order to clear an Alarm or Warning condition. For example: A Trip Point High value of 100 and a hysteresis value of 2 will result in an exception condition being set when the *Partial Pressure* is above 100 and cleared when the *Partial Pressure* drops below 98. Similarly, A Trip Point Low value of 100 and a hysteresis value of 2 will result in an exception condition being set when the *Partial Pressure* is below 100 and cleared when the *Partial Pressure* increases above 102.

### 5A-43.3 Common Services

**Table 5A-43.6 S-Partial Pressure Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 01 hex | n/a | Optional | Get_Attributes_All | Returns a predefined listing of this objects attributes (See the Get_Attributes_All Response definition below) |
| 08 hex | Optional | n/a | Create | Requests an instantiation of a new object within this class |
| 09 hex | Optional * | Optional | Delete | Deletes an object instance |
| 0E hex | Conditional** | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 10 hex | n/a | Required | Set_Attribute_Single | Modifies an attribute value. |

* The Delete service request at the class level deletes all instances.

**The Get_Attribute_Single service is REQUIRED if any attributes are implemented.

See Appendix A for definitions of these common services.

#### 5A-43.3.1 Get_Attributes_All Response

At the **Class level** this service is not supported.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5A-43.7 Get_Attributes_All Response Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | USINT | Number of Attributes | 0 |
| 2 | ARRAY of USINT | Attribute List | (null) |
| 3 thru 99 | See Note 1 | | |

Table Footnotes

1. Include the value of each attribute included in the Attribute List (attribute #2) in the order specified in that list. No default values apply for these attributes since only implemented attributes are included in Attribute List (attribute #2).

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-43.4 Object–specific Services

**Table 5A-43.8 S-Partial Pressure Object-specific Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 4B hex | Required | n/a | Create_Range | Requests a Create for range of instances.<br>See description below. |
| 4C hex | Required | n/a | Get_Instance_List | Requests a list of all enabled object instances with their respective AMU numbers |
| 4D hex | Optional | n/a | Get_Pressures | Requests a list of all enabled Partial Pressures above their respective thresholds |
| 4E hex | Optional | n/a | Get_All_Pressures | Requests a list of all enabled Partial Pressures |
| 4F hex | Optional | n/a | Group_Enable | Enables or Disables a group of instances as identified by their respective Group ID attribute values |

### 5A-43.4.1 Create_Range Service

**Table 5A-43.9 Create_Range Request Service Data Field Parameters**

| Parameter | Required | Data Type | Description |
|---|---|---|---|
| Start AMU | Required | REAL | Specifies the AMU value to be assigned to the first instance |
| End AMU | Required | REAL | Specifies the AMU value to be assigned to the last instance |
| Number of Instances | Required | UINT | Specifies the number of instances to be created |
| Starting ID | Required | UINT | Specifies the first instance ID in the list to be created.<br>0 = use next available |
| Group ID | Optional | USINT | Specifies the Group value to assign to the created instances. |

The successful response to a Create_Range service request is the creation of multiple S-Partial Pressure object instances. The device automatically sets the AMU value of each instance such that the range of AMU specified in the request is covered in evenly spaced increments by the specified number of instances.

If the Starting ID parameter value is zero (0), the device will start with the first available instance ID that allows for the Number of Instances specified to be created in one contiguous ID numbered range.

**Table 5A-43.10 Create_Range Success Response Service Data Field Parameters**

| Parameter | Data Type | Description |
|---|---|---|
| Starting ID | UINT | Indicates the first instance ID in the list created |
| Ending ID | UINT | Indicates the last instance ID in the list created |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-43.4.2 Get_Instance_List Service

The Get_Instance_List service request has no service data field parameters.

**Table 5A-43.11 Get_Instance_List Success Response Service Data Field Parameters**

| Parameter | Required | Data Type | Description | Semantics of Values |
|---|---|---|---|---|
| Size of List | Required | UINT | Specifies the number of elements in the Array | Number of object instances in the list |
| List of Gas Calibrations | Required if Size > 0 | ARRAY of | Supported List | The list of enabled object instances |
| | | STRUCT of | | |
| | | UINT | Instance Number | |
| | | USINT | Instance Type | |
| | | REAL | AMU | |
| | | REAL | Ending AMU | |
| | | UINT | Gas Standard Number | |

### 5A-43.4.3 Get_Pressures Service

The Get_Pressures service request has no service data field parameters.

**Table 5A-43.12 Get_Pressures Success Response Service Data Field Parameters**

| Parameter | Required | Data Type | Description | Semantics of Values |
|---|---|---|---|---|
| Size of List | Required | UINT | Specifies the number of elements in the Array | Number of Partial Pressures in the list |
| List of Gas Calibrations | Required if Size > 0 | ARRAY of | Supported List | The list of all enabled Partial Pressures above their respective thresholds |
| | | STRUCT of | | |
| | | UINT | Instance ID | |
| | | REAL | Partial Pressure | |

### 5A-43.4.4 Get_All_Pressures Service

The Get_All_Pressures service request has no service data field parameters.

**Table 5A-43.13 Get_All_Pressures Success Response Service Data Field Parameters**

| Parameter | Required | Data Type | Description | Semantics of Values |
|---|---|---|---|---|
| Size of List | Required | UINT | Specifies the number of elements in the Array | Number of Partial Pressures in the list |
| List of Gas Calibrations | Required if Size > 0 | ARRAY of | Supported List | The list of all enabled Partial Pressures |
| | | STRUCT of | | |
| | | UINT | Instance ID | |
| | | REAL | Partial Pressure | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-43.4.5 Group_Enable Service

**Table 5A-43.14 Group_Enable Request Service Data Field Parameters**

| Parameter | Required | Data Type | Description |
|---|---|---|---|
| Enable | Required | BOOL | 0 = Disable<br>1 = Enable |
| Group ID | Required | USINT | Specifies the Group (i.e., all S-Partial Pressure object instances whose Group ID attribute has this value is effected by this service request)<br>An unrecognized Group parameter returns an Invalid Parameter error. |

The Group_Enable Service Reply has no service data field parameters.

## 5A-43.5 Object Behavior

The behavior of this object is managed by the S-Device Supervisor Object, defined in Section 5A-35.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-44 S-Sensor Calibration Object

**Class Code: 40 Hex**

An S-Sensor Calibration Object affects the behavior of an associated S-Analog Sensor object instance; a device profile will show a relationship between these two objects where they are used. The S-Analog Sensor object uses the Calibration Object Instance attribute (Attribute ID 35) as a selection mechanism. The S-Sensor Calibration Object provides the data with which a device enacts the appropriate calibration algorithm for a given application. Usually, these calibration objects are instantiated in correspondence to various application parameters for a particular sensor type, for example fluid type. Each S-Sensor Calibration Object Instance contains a set of attribute values for one particular calibration set.

The S-Sensor Calibration class level object provides a service for retrieving a list of all valid object instances. The service response includes a list of elements. Each element includes: Instance ID, Calibration ID Number and the valid S-Analog Sensor object instance ID for which the instance is valid.

There may be more than one instance with the same Calibration ID Number. These instances may be differentiated by Full Scale, Calibration Name, Additional Scaler and/or other parametric distinctions, including valid S-Analog Sensor object instance ID. The distinctions may, or may not, be evident in the Get_All_Instances service response, depending upon what the distinction is.

S-Sensor Calibration Objects most often utilize the region of Manufacturer Defined Coefficients Attributes (ID 16-31) for specific calibration parameters. If more coefficients are needed, then they may be placed into the area of ID >100 – the vendor-specific range.

This object is a member of the *Hierarchy of Semiconductor Equipment Devices*. As such, its behavior is managed by the S-Device Supervisor Object. See Section 0 of this volume for more information.

## 5A-44.1 Class Attributes

**Table 5A-44.1 S-Sensor Calibration Object Class Attributes**

| Attribute ID | Need in implementation | Access Rule | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are either optional or conditional and are described in Chapter 5 of this specification. | | | | |
| 97-98 | Reserved by CIP for Future Use | | | | |
| 99 | Conditional * | Get | Subclass | UINT | Identifies a subset of additional class attributes, services and behaviors. |

* If the value of Subclass is 00, which identifies "no subclass", then this attribute is OPTIONAL in implementation, otherwise, this attribute is REQUIRED.

## 5A-44.2 Subclasses

Each class level subclass defines a unique meaning for an overlapping range of class attribute IDs and/or class service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given class is identified by the value of its Subclass class attribute. The subclasses for this object are specified at the end of this object specification section.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-44.3 Instance Attributes

Certain minimal implementations may support any optional "Set" attributes as "Get" only and still be compliant with this object specification. All required attributes must be supported as specified.

**Table 5A-44.2 S-Sensor Calibration Object Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Optional | Get | NV | Number of Attributes | USINT | Number of attributes supported | |
| 2 | Optional | Get | NV | Attribute List | ARRAY OF USINT | List of attributes supported by this object instance | |
| 3 | Required | Get | NV | Calibration ID Number | UINT | Identifies the Calibration (e.g., fluid type) | see Semantics section<br>[default] = 0 |
| 4 | Required | Get | NV | Valid Sensor Instance | UINT | S-Analog Sensor object instance ID for which this object instance is valid | 0 = No Valid Sensor<br>n = Instance ID<br>see Semantics section<br>[default] = 0 |
| 5 | Optional | Set | NV | Calibration Name | SHORT STRING | ASCII Text, max 50 characters, representation of the calibration (e.g. material type) | see Semantics section<br>[default] = null |
| 6 | Optional | Get | NV | Full Scale | STRUCT of: | Full Scale of the device using this object instance | see Semantics section<br>[default] = 0, 0 |
| | | | | | REAL | Amount | The amount of measured parameter corresponding to full scale. |
| | | | | | ENGUNITS | Units | The units for the above.<br>see Data Units Appendix D |
| 7 | Optional | Set | NV | Additional Scaler | REAL | Additional Correction Factor | In addition to the correction algorithm, this amount is multiplied to the reading.<br>Default = 1.0 |
| 8 | Optional | Get | NV | Calibration Date | DATE | Date of Calibration | The date this object instance was last calibrated<br>[default] = 0 |
| 9 | Optional | Set | NV | Temperature Data Units | ENGUNITS | Determines the Units context of the Temperature attribute | see Semantics section |
| 10 | Optional | Set | NV | Temperature | REAL | Temperature | see Semantics section |
| 11 | Optional | Set | NV | Pressure Data Units | ENGUNITS | Determines the Units context of the Pressure attribute | see Semantics section |
| 12 | Optional | Set | NV | Pressure | REAL | Pressure | see Semantics section |
| 13 | Optional | Set | NV | Coefficient A | REAL | The "a" term, in a quadratic correction | $y = ax^2 + bx + c$ |
| 14 | Optional | Set | NV | Coefficient B | REAL | The "b" term, in a quadratic correction | $y = ax^2 + bx + c$ |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**S-Sensor Calibration Object, Class Code: $40_{Hex}$**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 15 | Optional | Set | NV | Coefficient C | REAL | The "c" term, in a quadratic correction | $y = ax^2 + bx + c$ |
| 16-31 | Optional | Set | NV | Manufacturer Defined Coefficients ** | ** | ** | ** |
| 95-96 | Defined by Subclasses below | | | | | | |
| 97-98 | Reserved by CIP for Future Use | | | | | | |
| 99 | Conditional * | Get | NV | Subclass | UINT | Identifies a subset of additional instance attributes, services and behaviors. | 0 = No subclass<br>1 – 65535 = Reserved |

* If the value of Subclass is 00, then this attribute is OPTIONAL in implementation, otherwise, this attribute is REQUIRED.

** These attributes (16-31) are set aside for manufacturer specified calibration coefficients. This is to encourage the limitation and placement of these to facilitate the user's ability to download, upload and otherwise verify these values.

## 5A-44.4 Subclasses

Each instance level subclass defines a unique meaning for an overlapping range of instance attribute IDs and/or instance service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given instance is identified by the value of its Subclass instance attribute. The subclasses for this object are specified at the end of this object specification section.

## 5A-44.5 Semantics

### 5A-44.5.1 Calibration ID Number

Used to identify a particular calibration using a standardized identifying number, for which the object instance is currently calibrated. See Instance Application Example below.

The actual coding of these values are specified in the Device Profile or vendor specific implementation where used. Since the actual attributes, and their context, for the parameterization of object instances for particular calibration types is beyond the scope of this standard (i.e., vendor specific) the Access Rule for this attribute has been specified as Get. Vendors may choose to specify an Access Rule of Set for this attribute.

### 5A-44.5.2 Valid Sensor Instances

This attribute specifies the S-Analog Sensor object instance for which the S-Sensor Calibration object instance is valid. An S-Sensor Calibration object instance will be valid for zero or one S-Analog Sensor object instances.

### 5A-44.5.3 Calibration Name

This optional attribute is a string coded representation of the name of the calibration for which the object instance has been configured. It is coded as a user defined text symbol or it is coded as defined in the Device Profile where used.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-44.5.4 Full Scale

This optional attribute identifies the amount of measured parameter (e.g., Flow Rate) corresponding to the Full Scale of the associated S-Analog Sensor object. A primary purpose for this attribute is to allow for simple S-Analog Sensor object implementations where the Value is reported in raw units; this attribute allows a mapping to engineering units.

For example, the Full Scale for a S-Sensor Calibration object may be 100 LPM, while the Full Scale for the associated S-Analog Sensor object may be 20,000 counts (i.e., S-Analog Sensor object Data Type = INT and Data Units = Counts).

### 5A-44.5.5 Temperature and Pressure

These optional attributes are used for Calibrations where a temperature and/or a pressure dependency is known. These attributes may be supported with "set access" to allow the user the configure the calibration object or to continuously feed the value from the network. Alternatively, these attributes may be supported with "get access" to identify the temperature and/or pressure for which the calibration has been set. Another alternative is that these attributes may be shown with a linkage in a Device Profile whereby the value is modified internal to the device and would again be shown with "get access" to the network. In the latter case, the related Data Units attributes may not be supported.

### 5A-44.5.6 Instance Application Example

The following is an example to demonstrate the usage of S-Sensor Calibration object instances and their attributes:

A device has been supplied with three S-Sensor Calibration object instances: DI Water, Ethylene Glycol 50%, and Ethylene Glycol 70%. The user wishes to use the device for ethylene glycol 80% and knows that a correction factor of 1.05 will properly convert the ethylene glycol 70% calibration for this application. The object instance for ethylene glycol 70% would be selected and the *Additional Scaler* attribute for this instance would be set to 1.05. To identify this modification, the *Calibration Name* may be set to read “Ethylene Glycol 80%”, “HOCH2CH2OH”, or perhaps, “Coolant X”.

## 5A-44.6 Common Services

The S-Sensor Calibration Object provides the following Common Services:

**Table 5A-44.3 S-Sensor Calibration Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0Ehex | Conditional* | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 10hex | Conditional** | Required | Set_Attribute_Single | Modifies an attribute value. |

* Required if any attributes are supported.

** Required if any settable attributes are supported.

See Appendix A for definition of these services


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-44.7 Object–specific Services

**Table 5A-44.4 S-Sensor Calibration Object-specific Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 4Bhex | Required | n/a | Get_All_Instances | Requests a list of all available object instances with their respective Calibration ID Numbers |

### 5A-44.7.1 Get_All_Instances Request Service Data Field Parameters

None.

### 5A-44.7.2 Get_All_Instances Success Response

**Table 5A-44.5 S-Get_All_Instances Success Response Service Data Field Parameters**

| Parameter | Required | Data Type | Description | Semantics of Values |
|---|---|---|---|---|
| Size of List | Required | UINT | Specifies the number of elements in the Array | Number of calibrations in the list |
| List of Calibrations | Required if Size > 0 | ARRAY of | Supported List | The list of Sensor Calibrations |
| | | STRUCT of | Supported Calibration | |
| | | UINT | S-Sensor Calibration Object Instance ID | |
| | | UINT | Calibration ID Number | |
| | | UINT | Valid Sensor Instance | |

## 5A-44.8 S-Sensor Calibration Object Behavior

The behavior of this object is managed by the S-Device Supervisor Object, defined in Section 5A-35.6.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5A-45 Event Log Object

## Class Code: 41 Hex

The Event Log Object provides event indication and/or event logging on CIP nodes. Some applications may only require event indication, while others may only require event logging.

Event information is stored in arrays and these arrays have an assumed index equal to the element number in the array.

Multiple object instances are only required for devices which support independent logs. Each Event Log Object instance has an event instance identifier (user-defined number and optional name).

Each Event Log Object instance maintains a unique event log, which may contain time-stamped application data for that set of events. Each event associated with an instance may be assigned a unique identifier. An attribute of the Event Log Object Instance defines all of the events associated with the particular instance. Another attribute indicates the current state of each event. An event log entry occurs upon a state change in the status of any single event in the set of events being monitored by the particular Event Log Object Instance. Data stored within the event log is configurable to include application-specific data and time / date, in addition to the event identifier. The application data is optionally sent along with the event trigger from the application object to the Event Log Object. Both the event log size and its storage trigger mechanism are configurable.

The Event Log Object class maintains a list of the Event Log Object Instances.

The cause of an event is not defined for the Event Log Object. A local application is the event source, and the cause is application-specific. The application object sends the event trigger (typically Event Active indication) to the Event Log Object.

## 5A-45.1 Event Control

Events can be enabled and disabled as needed. When an event is disabled, all event triggers are ignored, meaning no log entries or occur. Events can also be silenced. A silenced event continues to log event information.

**Important:** Instances of the Event Log Object are divided into the following ranges.

**Table 5A-45.1 Event Log Instance ID Ranges**

| Range | Meaning | Quantity |
|---|---|---|
| 01 - 63 hex | Public | 99 |
| 64 hex - C7 hex | Vendor Specific | 100 |
| C8 hex - FF hex | Reserved by CIP for future use | 56 |
| 100 hex - 2FF hex | Public | 512 |
| 300 hex - 4FF hex | Vendor Specific | 512 |
| 500 hex - FFFF hex | Reserved by CIP for future use | 64,256 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-45.2 Class Attributes

**Table 5A-45.2 Event Log Object Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 thru 2 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | | |
| 3 | Conditional [3] | Get | NV | Number of Instances | UINT | Number of object instances currently created at this class level of the device | The number of object instances at this class hierarchy level. |
| 4 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | | |
| 32 | Conditional | Set | NV | Time Format | USINT | Data type code for time and/or date | 0xCD = DATE[1]<br>0xCE = TOD[1]<br>0xCF = DATE AND TIME[1]<br>0xD6 = FTIME[2]<br>0xD7 = LTIME[2]<br>0xD8 = ITIME[2]<br>0xDB = TIME[2] (Default) |
| 33 | Conditional | Set | NV | Present Time | Dependant on Attribute #32 | | Default time value = zero for all data types |
| 34 | Optional | Get | NV | Instance List | ARRAY of STRUCT of | List of all Instance numbers and names | |
| | | | | Instance number | UINT | Number of Event Log Object instance | |
| | | | | Instance name | STRINGI | Length and name of Event Log Object instance | |

Table Footnotes:
1 These data types result in a zero value on power-up/reset unless special hardware exists
2 The signed time data types shall not “roll over” into negative values; these values shall “roll over” to zero
3 This attribute is required when attribute #34 is implemented.

### 5A-45.2.1 Time Format

Defines the time format used for logging events. All Event Log Object instances shall use the same time format.

If time stamping is implemented, this attribute is required. If time stamping is not supported, this attribute shall not be implemented.

The time formats TIME OF DAY, DATE and DATE AND TIME, if implemented, require the device to continue advancing time across power cycling/reset actions.

Not all time formats are required to be supported. The time format “TIME” is required. If the implementation only supports the default time format “TIME”, this attribute is not required to be settable.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-45.2.2 Present Time

This attribute exposes a real time counter whose unit of measure is determined by the ***Time Format*** attribute. If time stamping is enabled in the Data Log, the value in this attribute is the source of the time stamp information.

If time stamping is supported, this attribute is required. If time stamping is not supported, this attribute shall not be implemented.

If the Set service is implemented for this attribute, the entire range of values, for the time format in effect, shall be supported for the Set service.

Attribute value roll over behavior: For any of the time formats that allow a negative value, the Present Time attribute shall not roll over into negative values.

### 5A-45.2.3 Instance List

The number(s) and name(s) of all instances contained in the Event Log Class.

## 5A-45.3 Instance Attributes

**Table 5A-45.3 Event Log Instance Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Optional | Set [3] | NV | Instance Name | STRINGI | Name of this event instance | |
| 2 | Required | Get | V | State | USINT | State of this instance. | 0 = Nonexistent<br>1 = Stopped<br>2 = Empty<br>3 = Available<br>4 = Full/Overwrite<br>5 = Full/Halted<br>6 – 255 = Reserved |
| 3 | Conditional[1] | Get | NV | Event List Size | UDINT | Number of elements in the Event List, Event Enable, Event Silenced and Event State attribute arrays. | See semantics |
| 4 | Optional | Set | NV | Event List | ARRAY of STRUCT | All event identifiers defined for this instance. | Event Identifier format is defined by attribute 24. |
| 5 | Optional | Set | NV | Event Enable | ARRAY of BOOL | Enables each event | 0 = Disabled<br>1 = Enabled<br>Default = Enabled |
| 6 | Optional | Set | NV | Event Silenced | ARRAY of BOOL | Inhibits selected Event from continuing to signal it's presence in the Event Log as an active event, regardless of it's present condition (Active/Inactive, Logged/Not Logged) | 0 = Not silenced<br>1 = Silenced<br>Default = Not silenced |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Event Log Object, Class Code: $41_{Hex}$

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 7 | Optional | Get | V | Event State | ARRAY of BYTE | Identifies the current state of each event | Bit 0 (See attribute #5)<br>0 = Disabled<br>1 = Enabled<br>Bit 1<br>0 = Inactive<br>1 = Active<br>Bit 2<br>0 = UnAcked<br>1 = Acked<br>Bit 3<br>0 = Not Logged<br>1 = Logged<br>Bit 4 (See attribute #6)<br>0 = Not Silenced<br>1 = Silenced<br>Bits 5 7<br>(reserved)<br>Default = 0x0B |
| 8 | Optional | Set | NV | Logged States Configuration | BYTE | Configures which event state transitions are logged | See Semantics below.<br>Default = 0x01 |
| 9 | Optional | Set | NV | Logged Data Configuration | BYTE | Configures which data is stored in the event log. | See Semantics below.<br>Default = 0 |
| 10 | Optional | Set | NV | Log Full Action | USINT | Configures the action to take when a new event is detected and the log is full. | 0 = Halt (Default)<br>1 = Scroll<br>2 – 255 = Reserved |
| 11 | Optional | Set | NV | Duplicate Event Action | USINT | Configures the action to take when a duplicate event is detected. | 0 = Ignore (Default)<br>1 = Add<br>2 = Overwrite<br>3 – 255 = Reserved |
| 12 | Required | Get | NV | Event/Data Log Maximum Size | UDINT | Maximum number of allowable entries in Event/Data Log. | |
| 13 | Required | Get | V | Event/Data Log Size | UDINT | The present number of entries in the Event/Data Log. | 0 to Event Log Maximum Size |
| 14 | Required | Set | V[2] | Event/Data Log | ARRAY of STRUCT | List of all events that have been logged | See semantics |
| 15 | Optional | Get | V | Active Event/Data List Size[1] | UDINT | Number of entries in Active Event List | 0 to Event List Size |
| 16 | Optional | Get | V | Active Event/Data List | ARRAY of STRUCT | List of event identifiers that are presently active | See semantics |
| 17 | Optional | Get | V | Logged Event/Data List Size[1] | UDINT | Number of entries in Logged Event List | 0 to Event Log Maximum Size |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Event Log Object, Class Code: $41_{Hex}$**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 18 | Optional | Get | V | Logged Event/Data List | ARRAY of STRUCT | List of event identifiers that are currently logged | See semantics |
| 19 | Optional | Get | V | Log Full | BOOL | TRUE if Event Log Size = Event Log Maximum Size | 0= Log not full<br>1 = Log full |
| 20 | Optional | Get | V | Log Contains Entries | BOOL | Indicates if there are any events in the log. | 0 = Log empty<br>1 = Log contains entries |
| 21 | Optional | Get | V | Log Overrun | BOOL | Indicates if there has been an overrun of the log. | 0 = No Log Overrun<br>1 = Log Overrun |
| 22 | Optional | Set | V | Sequential Event/Data Access | See Attribute 9 | Provides a simple mechanism to access Event/Data entry | See Semantics |
| 23 | Optional | Set | NV | Startup Behavior | USINT | Specifies the behavior of the instance after Power-up or Reset | 0= Auto Active (Default)<br>1= Start Required<br>2-255 = Reserved |
| 24 | Optional | Set | NV | Event Identifier Format | USINT | Defines the format of the Event Identifier for the log entries. | See Semantics<br>Default = 0 |

Table Footnotes:
1 This attribute indicates the size of a 'list' provided by another attribute and is required if that 'list' attribute is present.
2 The logged data is optionally nonvolatile.
3 If dynamic instances are possible (via the Create service) and this attribute is implemented, the SetAttributeSingle service is required.

## 5A-45.4 Semantics:

### 5A-45.4.1 Attribute #1 - Instance Name

Further defines the instance by permitting a string name for the event instance.

### 5A-45.4.2 Attribute #2 - State

See State Transition Diagram (STD) and State Event Matrix (SEM) in section 5A-45.6 below.

### 5A-45.4.3 Attribute #3 - Event List Size

The number of members in the Event List, Event Enabled, Event Silenced and Event State attributes.

### 5A-45.4.4 Attribute #4 - Event List

The list of unique event identifiers recorded by this object instance. This may not include all events which could be logged.

### 5A-45.4.5 Attribute #5 - Event Enable

Enables or disables the event completely with regard to the Event Log Object. Default is enabled

### 5A-45.4.6 Attribute #6 - Event Silenced

Disables the updating of the selected Event regardless of the state of the event, which allows the user to stop transmission of individual event indication messages.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-45.4.7 Attribute #7 - Event State

Current event state for each of the events. Refer to Object Behavior section.

### 5A-45.4.8 Attribute #8 - Logged States Configuration

Defines when an event log entry is recorded, based on event state changes. Allows the user to select which state changes require an event log entry, optimizing event log memory usage. More than one state change can be enabled simultaneously. Selecting no state changes disables event logging without disabling the state monitoring capability of the object.

**Table 5A-45.4 Defined Values For the Logged States Configuration Attribute**

| Value | Description |
|---|---|
| Bit 0 | Log state transitions to Active or Inactive |
| Bit 1 | Log state transitions to Disabled or Enabled |
| Bit 2 | Log state change due to UserAck |
| | |
| Bits 3 - 7 | Reserved |

### 5A-45.4.9 Attribute #9 - Logged Data Configuration

Defines the information to be stored in the Event/Data Log (Attribute 14). The logged data shall always begin with the Event Identifier (format defined in Attribute 24), and may be followed by optional data blocks as they appear in the table below, from low bit (Bit 0) to high bit (Bit 7). If a bit is set, the corresponding data block appears in the Event/Data Log.

**Table 5A-45.5 Defined Values For the Logged Data Configuration Attribute**

| Value | Description |
|---|---|
| Bit 0 | Time value (data value defined by Time Format class attribute) |
| Bit 1 | New event state (data value defined by Event State attribute) |
| Bit 2 | Application data1 |
| Bits 3 - 7 | Reserved |

Table Footnotes:

1. The Application data shall be in the form of a Data Segment (see Appendix C)

### 5A-45.4.10 Attribute #10 - Log Full Action

Defines the action to take when a new event is detected and the event log is full. Options include scroll and halt. The scroll selection causes new event log entries to overwrite the oldest entries when the event log is full. The halt selection causes all new event log entries to be lost when the event log is full.

**Table 5A-45.6 Defined Values For the Log Full Action Attribute**

| Value | Description |
|---|---|
| 0 | Halt (stop when log is full) |
| 1 | Scroll (oldest entry is deleted, member list is shuffled down, and newest entry is last in the list) |
| 2 – 255 | Reserved |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-45.4.11 Attribute #11 - Duplicate Event Action

Defines the action to take when a duplicate event is detected. Options include ignore, add, and overwrite. The ignore option causes event entries of the same Event Identifier to be discarded. The overwrite option causes event entries of the same event to overwrite the previous entry of that event. The add to log option allows the same event to be placed in the log each time it occurs.

**Table 5A-45.7 Defined Values For the Duplicate Event Action Attribute**

| Value | Description |
|---|---|
| 0 | Ignore |
| 1 | Add |
| 2 | Overwrite |
| 3 – 255 | Reserved |

### 5A-45.4.12 Attribute #12 - Event/Data Log Maximum Size

The maximum number of allowable event/data log entries for the instance. This number may vary based on available memory and how the user has configured the event log options.

### 5A-45.4.13 Attribute #13 - Event/Data Log Size

The current number of non-zero entries in an instance's event/data log.

### 5A-45.4.14 Attribute #14 - Event/Data Log

This attribute is the event/data log for the instance. New events are added to the end of the log, subject to overriding behavior specified by the Log Full Action and Duplicate Event Action attributes.

Information contained in the event/data log depends upon its configuration. An entry whose value is all zero is defined to be a null entry containing no useful data. This attribute is a single dimension array of entries. The maximum size of the array is given by attribute #12 (Event/Data Log Maximum Size). The present number of entries is given by attribute #13 (Event/Data Log Size). The form of each entry is determined by the combination of the Event Identifier Format (attribute #24) in conjunction with optional data selected by the Logged Data Configuration (attribute #9). The table below shows the structure of a logged event entry.

**Table 5A-45.8 Structure of Event Entry**

| Field Name | Need in Entry | Where enabled | Where Format Defined |
|---|---|---|---|
| Event Identifier | Required | Always Present | Event Identifier Format (Instance Attribute 24) |
| Time Stamp | Optional | Logged Data Configuration (Instance Attribute 9), Bit 0 | Time Format (Class Attribute 32) |
| Event State | Optional | Logged Data Configuration (Instance Attribute 9), Bit 1 | Event State (Instance Attribute 7) |
| Application Data | Optional | Logged Data Configuration (Instance Attribute 9), Bit 2 | Encoded Data Segment as defined in Appendix C |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

For example, assuming the Log Entry Form attribute is the value zero, then the first four octets of each member entry shall conform to the 32-bit object model/error response format (the Event Identifier). If any optional data is selected by the Logged Data Configuration attribute, the selected data will be present in the logged event entry following the Event Identifier. The order of the selected fields is defined by the Logged Data Configuration attribute and shown in the table above. Continuing this example, assuming the value 0x03 exists in the Logged Data Configuration attribute and class attribute #32 (Time Format) is the value 0xDB (TIME data type), the following table shows the form of each Event/Data Log member entry:

**Table 5A-45.9 Event/Data Log Member Entry Form**

| Byte Offset | Contains | |
|---|---|---|
| 0 | Event Identifier | 8-bit Class Code |
| 1 | | 8-Bit Instance |
| 2 | | 8-Bit Error/Event Code |
| 3 | | 8-Bit Extended Event/Error Code |
| 4 | Time Value | Time Stamp for entry (Low Order Byte) |
| 5 | | Time Stamp (next byte) |
| 6 | | Time Stamp (next byte) |
| 7 | | Time Stamp (High Order Byte) |
| 8 | New Event State | Event State |

### 5A-45.4.14.1 Service Behavior:

#### 5A-45.4.14.1.1 Get_Attribute_Single

Returns the entire content of the attribute. Each member of the Event Log is returned, regardless of whether the content of the member is non-null (valid). Therefore, this service always returns “Event Log Maximum Size” times each entry’s “form size” bytes.

**Table 5A-45.10 Get_Attribute_Single Service Error Code**

| Error Code | Error Name | Description |
|---|---|---|
| 0x15 | Too Much Data | The present size of the message necessary to transmit all Event Log members exceeds the messaging capacity of the node |

#### 5A-45.4.14.1.2 Set_Attribute_Single

The Set_Attribute_Single service shall not be implemented

**Table 5A-45.11 Set_Attribute_Single Service Error Code**

| Error Code | Error Name | Description |
|---|---|---|
| 0x08 | Service Not Supported | The Set service is not supported for this attribute |

#### 5A-45.4.14.1.3 Get_Member

Returns the content of the specified member within the Event/Data Log

**Table 5A-45.12 Get_Member Service Error Codes**

| Error Code | Error Name | Description |
|---|---|---|
| 0x28 | Invalid Member ID | Member number zero was selected |
| 0x08 | Service Not Supported | This implementation of the Event Log object does not support the Get Member service |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-45.4.14.1.4 Set_Member

**Table 5A-45.13 Set_Member Service Error Codes**

| Error Code | Error Name | Description |
|---|---|---|
| 0x28 | Invalid Member ID | The specified member does not presently exist |
| 0x15 | Too Much Data | The service data of the request contained too much data for the form of the Event Log |
| 0x13 | Not Enough Data | The service data of the request did not contain enough data for the form of the Event Log |
| 0x0C | Object State Conflict | The state of the instance is any of:<br>Stopped<br>Halted<br>or the specified member is valid and cannot be overwritten |
| 0x09 | Invalid Attribute Value | The value of the data within the request was rejected by the Event Log |
| 0x08 | Service Not Supported | This implementation of the Event Log object does not support the Set Member service |

### 5A-45.4.14.1.5 Remove_Member

Returns the content of the specified member within the Event/Data Log and then the member is removed from the Log.

**Table 5A-45.14 Remove_Member Service Error Codes**

| Error Code | Error Name | Description |
|---|---|---|
| 0x28 | Invalid Member ID | The specified member does not exist |
| 0x08 | Service Not Supported | This implementation of the Event Log object does not support the Remove Member service |

### 5A-45.4.14.1.6 Insert_Member

Inserts the content of the service data into the Event/Data Log at the specified member position.

**Table 5A-45.15 Insert_Member Service Error Codes**

| Error Code | Error Name | Description |
|---|---|---|
| 0x28 | Invalid Member ID | The specified member does not exist |
| 0x15 | Too Much Data | The service data of the request contained too much data for the form of the Event Log |
| 0x13 | Not Enough Data | The service data of the request did not contain enough data for the form of the Event Log |
| 0x0C | Object State Conflict | The state of the instance is any of:<br>Stopped<br>Halted<br>or the specified member is valid and cannot be moved in the member list |
| 0x09 | Invalid Attribute Value | The value of the data within the request was rejected by the Event Log |
| 0x08 | Service Not Supported | This implementation of the Event Log object does not support the Insert Member service |

### 5A-45.4.15 Attribute #15 - Active Event List Size

The current number of entries in the Active Event List.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-45.4.16 Attribute #16 - Active Event List

A list of all currently active events related to the instance from which the list is obtained.

### 5A-45.4.17 Attribute #17 - Logged Event List Size

The current number of entries in the Logged Event List. This list shall be the same size as the Event List.

### 5A-45.4.18 Attribute #18 - Logged Event List

A list of all currently logged events related to the instance from which the list is obtained.

### 5A-45.4.19 Attribute #19 - Log Full

A flag which indicates if the Event Log is full. TRUE if Event Log Size = Event Log Maximum Size, and intended primarily for dynamic log management.

### 5A-45.4.20 Attribute #20 - Log Not Empty

A flag which indicates that the Event Log is not empty (ie. at least one entry is present). This value is TRUE when the Event Log Size is not equal to zero.

### 5A-45.4.21 Attribute #21 - Log Overrun

A flag which indicates that an overrun has occurred. This value is reset to zero (FALSE) when the attribute is read.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-45.4.22 Attribute #22 – Sequential Event/Data Access

The Sequential Event/Data Access attribute provides the means for simple sequential access to members of the Event/Data Log. The form of the data in this attribute is identical to the form of the data contained within individual member(s) of the Event/Data Log (attribute #14). The behavior of services to the Sequential Event/Data Access attribute is as follows:

**Table 5A-45.16 Sequential Event/Data Access Behavior**

| Sequential Event/Data Access (attribute #22) | | Results in Service to Event/Data Log (attribute #14) | | |
|---|---|---|---|---|
| **Service Code** | **Service Name** | **Service Code** | **Service Name** | **Behavior** |
| 0x0E | Get_Attribute_ Single | 0x1B | Remove_Member | The Get_Attribute_Single service, when directed to the Sequential Event/Data Access attribute, causes a Remove_Member service to be directed to the Event/Data Log attribute of the Event Log object instance. The internally generated Remove_Member service will always attempt to remove member one of the Event/Data Log attribute.<br>If the Event/Data log is empty, this service shall return no data.<br>If the Sequential Event/Data Access attribute is implemented, support for the Remove_Member service is required. |
| 0x10 | Set_Attribute_ Single | 0x1A | Insert_Member | The Set_Attribute_Single service, when directed to the Sequential Event/Data Attribute , causes an Insert_Member service to be directed to the Event/Data Log attribute of the Event Log object instance. The internally generated Insert_Member service will always attempt to insert the last member of the Event/Data Log attribute.<br>If the Set_Attribute_Single service is implemented for this attribute support for the Insert_Member service is required. |

### 5A-45.4.23 Attribute #23 - Startup Behavior

Defines the behavior of the instance state machine at startup and reset.

**Table 5A-45.17 Startup Behavior Attribute Values**

| Value | Description |
|---|---|
| 0 | Auto Active |
| 1 | Start Required |
| 2 – 255 | Reserved |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5A-45.4.24 Attribute #24 - Event Identifier Format

Defines the format of the Event Identifier.

**Table 5A-45.18 Values for Attribute #24**

| Value | Description |
|---|---|
| 0 | 32-bit object model/error format |
| 1 | 48-bit object model/error format |
| 2 | 64-bit object model/error format |
| 3 | 16-bit unique identifier value |
| 4 – 254 | Reserved |
| 255 | Product specific – fixed form |

**Table 5A-45.19 32-Bit Object Model / Error Format**

| Byte Offset | Data Type | Description |
|---|---|---|
| 0 | USINT | 8-bit object class identification |
| 1 | USINT | 8-bit object instance identification |
| 2 | USINT | 8-bit Error/Event code |
| 3 | USINT | 8-bit Extended Error Code |

**Table 5A-45.20 48-Bit Object Model / Error Format**

| Byte Offset | Data Type | Description |
|---|---|---|
| 0 | UINT | 16-bit object class identification |
| 2 | USINT | 8-bit object instance identification |
| 3 | USINT | 8-bit Error/Event code |
| 4 | UINT | 16-bit Extended Error Code |

**Table 5A-45.21 64-Bit Object Model / Error Format**

| Byte Offset | Data Type | Description |
|---|---|---|
| 0 | UINT | 16-bit object class identification |
| 2 | UINT | 16-bit object instance identification |
| 4 | UINT | 16-bit Error/Event code |
| 6 | UINT | 16-bit Extended Error Code |

**Table 5A-45.22 16-Bit Unique Identifier Value**

| Byte Offset | Data Type | Description |
|---|---|---|
| 0 | UINT | 16-bit unique identifier |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-45.5 Common Services

**Table 5A-45.23 Common Services for the Event Log Object**

| Service Code | Need In Implementation | | Service Name | ServiceDescription |
|---|---|---|---|---|
| | Class | Instance | | |
| $05_{hex}$ | N/A | Required | Reset | Provides one of the types of reset for the object as described below. |
| $06_{hex}$ | N/A | Conditional[1] | Start | Transitions the object from the Configuring state to the Active state, which allows event logging to commence. |
| $07_{hex}$ | N/A | Conditional[1] | Stop | Transitions the object from the Active state to the Configuring state. This service causes the event list and status attributes to be cleared and allows for the setting of some attributes which are get-only in the Active state. |
| $0E_{hex}$ | Optional[2] | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| $10_{hex}$ | Optional | Optional | Set_Attribute_Single | Sets the contents of the specified attribute. |
| $18_{hex}$ | Optional | Optional | Get_Member | Returns the contents of a member of the specified attribute. |
| $19_{hex}$ | Optional | Optional | Set_Member | Modifies a member of the specified attribute. |
| $1A_{hex}$ | N/A | Optional | Insert_Member | Adds a member to one of the list attributes. |
| $1B_{hex}$ | N/A | Optional | Remove_Member | Removes a member from one of the list attributes. |

Table Footnotes

1 The Delete, Start and Stop services are required if the Create Service is supported.

2 The Get_Attribute_Single service is required if any attributes are implemented.

### 5A-45.5.1 Reset Service

The Reset service has the following object specific parameter:

**Table 5A-45.24 Reset Service Object Specific Parameters**

| Name | Data Type | Description of Parameter | Semantics of Values |
|---|---|---|---|
| Reset Type | USINT | Type of reset requested. | See table below. |

The parameter Reset Type for the Reset service has the following value specification:

**Table 5A-45.25 Reset Type Value Descriptions**

| Value | Description of Value |
|---|---|
| 0 | Emulate as closely as possible cycling power, returning all attributes to the power up values. This is the default if the parameter is omitted. |
| 1 | Return as closely as possible to the out-of-box configuration, then emulate cycling power as closely as possible. |
| 2 – 255 | Reserved. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5A-45.6 Event/Data Log Instance Behavior

**Figure 5A-45.1 Event/Data Log Instance State Transition Diagram**

**Table 5A-45.26 Event/Data Log Object State Event Matrix**

| | Event/Data Log Object Instance State Event Matrix | | | | | |
|---|---|---|---|---|---|---|
| **Event** | **Non-Existent (0)** | **Stopped (1)** | **Empty (2)** | **Available (3)** | **Full Overwrite (4)** | **Full Halted (5)** |
| Power Up | *If* set to "Power up active" transition to Empty<br>*Else* transition to Stopped. | Not Applicable | Not Applicable | Not Applicable | Not Applicable | Not Applicable |
| Create | Transition to Stopped | Not Applicable | Not Applicable | Not Applicable | Not Applicable | Not Applicable |
| Delete | Not Applicable | Delete all resources and transition to Non-Existent. | Return Error (Object State Conflict) | Return Error (Object State Conflict) | Return Error (Object State Conflict) | Return Error (Object State Conflict) |
| Start (Not supported) | Not Applicable | Return Error (Service not supported) | Return Error (Service not supported) | Return Error (Service not supported) | Return Error (Service not supported) | Return Error (Service not supported) |
| Start (Supported) | Not Applicable | Transition to Available | Return Error (Object Already in State/Mode) | | | |
| Stop (Not supported) | Not Applicable | Return Error (Service not supported) | Return Error (Service not supported) | Return Error (Service not supported) | Return Error (Service not supported) | Return Error (Service not supported) |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Event Log Object, Class Code: $41_{Hex}$

| | Event/Data Log Object Instance State Event Matrix | | | | | |
|---|---|---|---|---|---|---|
| **Event** | **Non-Existent (0)** | **Stopped (1)** | **Empty (2)** | **Available (3)** | **Full Overwrite (4)** | **Full Halted (5)** |
| Stop (Supported) | Not Applicable | Return Error (Object Already in State/Mode) | Clear the Event Log, Active Event List, Logged Event List. Set Log Full, Log Not Empty, and Log Overrun attributes to zero. Transition to Stopped. | | | |
| Reset (Type = 0 or 1) | Not Applicable | Perform Reset as defined by service. | Perform Reset as defined by service. | | | |
| Get Attribute | Not Applicable | Validate/service the request. Return response. | | | | |
| Set Attribute | Not Applicable | Validate/service the request. Return response. | | | | |
| SetAttribute | Not Applicable | Validate/service the request. Return response. | | | | |
| to "Log Full Action" attribute | | | | | If attribute value after service is zero (Halt) – Transition to Full Halted | If attribute value after service is one (Scroll) – Transition to Full Scroll |
| Insert Member (internal) | Not Applicable | Ignore | *If* event Not Enabled discard<br>*Else* process event based on Logged Data, Log Full Action, and Duplicate Event Action attributes. Update Log Full, Log Not Empty, and Log Overrun attributes accordingly. | | | Ignore |
| | | | Transition to Available state | If Log is now full and Log Full Action is "Halt" – Transition to "Full Halted"<br><br>If Log is now full and Log Full Action is "Scroll" – Transition to "Full Scroll" | | |
| Insert Member (via explicit service or the Sequential Event/Data Access attribute) | Not Applicable | Return error (Object State Conflict) | *If* event Not Enabled discard<br>*Else* process event based on Logged Data, Log Full Action, and Duplicate Event Action attributes. Update Log Full, Log Not Empty, and Log Overrun attributes accordingly. | | | Return error (Object State Conflict) |
| | | | Transition to Available state | If Log is now full and Log Full Action is "Halt" – Transition to "Full Halted"<br><br>If Log is now full and Log Full Action is "Scroll" – Transition to "Full Scroll" | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Event Log Object, Class Code: $41_{Hex}$**

| | Event/Data Log Object Instance State Event Matrix | | | | | |
|---|---|---|---|---|---|---|
| **Event** | **Non-Existent (0)** | **Stopped (1)** | **Empty (2)** | **Available (3)** | **Full Overwrite (4)** | **Full Halted (5)** |
| Remove Member | Not Applicable | Return error (Object State Conflict) | Return error (Object State Conflict) | Remove member at specified index. Update Log Full and Log Not Empty attributes accordingly. | | |
| | | | | If last entry removed, Transition to Empty | Transition to Available | Transition to Available |
| Get Member | Not Applicable | Validate/service the request. Return response. | | | | |

### 5A-45.6.1 Event Behavior

The Event Log object defines optional states and state transitions for events. This behavior definition does not define what physically causes the transition triggers. For example, UserACK/Silence could be a pushbutton on one device, and may require an HMI interactive sequence on another. The behavior of an event is illustrated in the State Transition Diagram and State Event Matrix below.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5A-45.2 Event State Transition Diagram**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5A-45.27 State Event Matrix for Events**

| Event Trigger | State | | | | |
|---|---|---|---|---|---|
| | **Disabled** | **Inactive-Acked** | **Active-Acked** | **Inactive** | **Active** |
| Disable | Ignore | Transition to Disabled | If not Silenced, Event State = Inactive-Acked<br>Transition to Disabled | If not Silenced, Event State = Inactive-Acked<br>Transition to Disabled | If not Silenced, Event State = Inactive-Acked<br>Transition to Disabled |
| Enable | Transition to Inactive-Acked | Ignore | Ignore | Ignore | Ignore |
| Silence | Ignore | Ignore | Ignore | Event State = Inactive-Acked<br>Transition to Inactive-Acked | Event State = Active-Acked<br>Transition to Active-Acked |
| Unsilence | Ignore | Event State = Inactive-Acked | Event State = Active-Acked | Not applicable | Not applicable |
| Activate (major) | Ignore | If not silenced, Event State = Active<br>Log event (if enabled)<br>If silenced, transition to Active-Acked<br>If not silenced, transition to Active | If not silenced, Event State = Active<br>Log event (if enabled)<br>If not silenced, transition to Active | If not silenced, Event State = Active<br>Log event (if enabled)<br>If silenced, transition to Active-Acked<br>If not silenced, transition to Active | Not applicable |
| Activate (minor) | Ignore | If not silenced, Event State = Active-Acked<br>Log event (if enabled)<br>Transition to Active-Acked | Ignore | Not applicable | Not applicable |
| Deactivate | Ignore | Ignore | If not silenced, Event State = Inactive-Acked<br>Transition to Inactive-Acked | Ignore | If not silenced, Event State = Inactive<br>If not silenced, transition to Inactive.<br>If silenced, transition to Inactive-Acked. |
| UserACK | Ignore | Ignore | Ignore | If not silenced, Event State = Inactive-Acked<br>Transition to Inactive-Acked | If not silenced, Event State = Active-Acked<br>Transition to Active-Acked |
| All Member Services (except Remove_ Member) | Service the request and return the appropriate response. | Service the request and return the appropriate response. | Service the request and return the appropriate response. | Service the request and return the appropriate response. | Service the request and return the appropriate response. |
| Remove_ Member | Service the request and return the appropriate response. | Service the request and return the appropriate response. | Return Object State Conflict error (0C hex) | Service the re- quest and return the appropriate response.<br>Transition to Inactive-Acked | Return Object State Conflict error (0C hex) |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This page is intentionally left blank


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Volume 1: Common Industrial Protocol (CIP™) Specification

# Chapter 5B: Object Library, Part B


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Contents


5B-1 Object Library Continued ..... 6
5B-2 Time Sync Object ..... 7
5B-2.1 Time Sync Object Overview ..... 7
5B-2.2 Revision History ..... 7
5B-2.3 Class Attributes ..... 7
5B-2.4 Instance Attributes ..... 8
5B-2.4.1 Attribute 1: PTPEnable ..... 11
5B-2.4.2 Attribute 2: IsSynchronized ..... 11
5B-2.4.3 Attribute 3: SystemTimeMicroseconds ..... 11
5B-2.4.4 Attribute 4: SystemTimeNanoseconds ..... 12
5B-2.4.5 Attribute 5: OffsetFromMaster ..... 12
5B-2.4.6 Attribute 6: MaxOffsetFromMaster ..... 12
5B-2.4.7 Attribute 7: MeanPathDelayToMaster ..... 12
5B-2.4.8 Attribute 8: GrandmasterInfo ..... 12
5B-2.4.9 Attribute 9: ParentClockInfo ..... 12
5B-2.4.10 Attribute 10: LocalClockInfo ..... 12
5B-2.4.11 Attribute 11: NumberOfPorts ..... 15
5B-2.4.12 Attribute 12: Port State ..... 15
5B-2.4.13 Attribute 13: PortEnableCfg ..... 16
5B-2.4.14 Attribute 14: PortLogAnnounceIntervalCfg ..... 16
5B-2.4.15 Attribute 15: PortLogSyncIntervalCfg ..... 16
5B-2.4.16 Attribute 16: Priority1 ..... 16
5B-2.4.17 Attribute 17: Priority2 ..... 16
5B-2.4.18 Attribute 18: DomainNumber ..... 16
5B-2.4.19 Attribute 19: ClockType ..... 17
5B-2.4.20 Attribute 20: ManufactureIdentity ..... 17
5B-2.4.21 Attribute 21: ProductDescription ..... 17
5B-2.4.22 Attribute 22: RevisionData ..... 17
5B-2.4.23 Attribute 23: UserDescription ..... 18
5B-2.4.24 Attribute 24: PortProfileIdentityInfo ..... 18
5B-2.4.25 Attribute 25: PortPhysicalAddressInfo ..... 18
5B-2.4.26 Attribute 26: PortProtocolAddressInfo ..... 18
5B-2.4.27 Attribute 27: StepsRemoved ..... 19
5B-2.4.28 Attribute 28: SystemTimeAndOffset ..... 19
5B-2.5 Common Services ..... 19
5B-2.5.1 Get_Attributes_All Response ..... 20
5B-2.6 Object-Specific Services ..... 20
5B-2.7 CIP Sync Overview ..... 20
5B-2.8 Definitions ..... 20
5B-2.9 Overview of the IEEE 1588 Precision Time Protocol (PTP) ..... 21
5B-2.10 CIP Sync System Time Definition ..... 23
5B-2.11 CIP Sync Implementation ..... 23
5B-2.11.1 PTP Clock Implementation ..... 23
5B-2.11.2 PTP Protocol Stack Implementation ..... 23
5B-2.12 1588 PTP Version support ..... 23
5B-2.13 Clock Identity ..... 23
5B-2.14 PTP Port Assignments ..... 24
5B-2.15 CIP Sync PTP Profile ..... 24
5B-2.15.1 Profile Identification ..... 24
5B-2.15.2 Profile Default Settings and Ranges ..... 25
5B-2.15.3 Profile Transports ..... 25
5B-2.15.4 PTP Domain ..... 25
5B-2.15.5 PTP Node management ..... 25



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

5B-2.15.6 Transparent Clock ..... 25
5B-2.15.7 Path delay mechanism ..... 25
5B-2.15.8 Recommended PTP Clock Settings ..... 26
5B-2.15.9 TTL – Time to Live (EtherNet/IP only) ..... 26
5B-2.15.10 Handset Clock Quality Management ..... 26
5B-2.15.11 Device Level Ring and Path Reconfiguration ..... 27
5B-2.16 CIP Sync Clock Model ..... 28
5B-2.17 CIP Sync System Time Compensation ..... 30
5B-2.17.1 Step Compensation Algorithms ..... 30
5B-2.18 CIP Sync Group Startup Sequence ..... 32
5B-2.19 Time Sync Status Indication ..... 33
5B-2.20 Instance Attribute Changes from revision 1 to revision 2 ..... 34
5B-3 Modbus Object ..... 37
5B-3.1 Class Attributes ..... 37
5B-3.2 Instance Attributes ..... 37
5B-3.3 Common Services ..... 38
5B-3.4 Object-specific Services ..... 38
5B-3.4.1 Read_Discrete_Inputs Service ..... 38
5B-3.4.2 Read_Coils Service ..... 39
5B-3.4.3 Read_Input_Registers Service ..... 39
5B-3.4.4 Read_Holding_Registers Service ..... 40
5B-3.4.5 Write_Coils Service ..... 40
5B-3.4.6 Write_Holding_Registers Service ..... 41
5B-3.4.7 Modbus_Passthrough Service ..... 41
5B-3.5 Error Checking ..... 42
5B-4 Originator Connection List Object ..... 43
5B-4.1 Class Attributes ..... 43
5B-4.2 Instance Attributes ..... 43
5B-4.3 Services Supported ..... 43
5B-4.3.1 Create (0x08) ..... 44
5B-4.3.2 Delete (0x09) ..... 45
5B-4.3.3 Connection Read (0x4C) ..... 45
5B-5 Target Connection List Object ..... 47
5B-5.1 Class Attributes ..... 48
5B-5.1.1 Connection Data Set Revision (Attribute 8) ..... 48
5B-5.1.2 Capabilities (Attribute 9) ..... 48
5B-5.2 Instance Attributes ..... 49
5B-5.3 Instance Assignments ..... 50
5B-5.3.1 Connectivity Test Connection - Instance #1 ..... 50
5B-5.4 Connection Points ..... 50
5B-5.5 Common Services ..... 51
5B-5.5.1 Get_Attributes_All Response ..... 51
5B-5.6 Object Specific Services Supported ..... 52
5B-5.6.1 Connection Read (0x4C) ..... 52
5B-6 Base Energy Object ..... 56
5B-6.1 Revision History ..... 56
5B-6.2 Class Attributes ..... 57
5B-6.3 Instance Attributes ..... 57
5B-6.3.1 Semantics ..... 59
5B-6.4 Common Services ..... 75
5B-6.4.1 Reset Service ..... 75
5B-6.4.2 Get_Attributes_All Response ..... 76
5B-6.4.3 Create Service ..... 77
5B-6.4.4 Delete Service ..... 78
5B-6.5 Object Specific Services ..... 79
5B-6.5.1 Start_Metering Service ..... 79


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

5B-6.5.2 Stop_Metering Service ..... 79
5B-6.6 Behavior ..... 79
5B-6.7 Aggregation Application Considerations ..... 81
5B-6.8 Modbus Integration Considerations (Odometers/Base Energy Object in a Modbus Translator) (Informative)84
5B-7 Electrical Energy Object ..... 85
5B-7.1 Revision History ..... 85
5B-7.2 Class Attributes ..... 85
5B-7.3 Instance Attributes ..... 86
5B-7.3.1 Semantics ..... 88
5B-7.4 Common Services ..... 91
5B-7.4.1 Get_Attributes_All Response ..... 92
5B-7.5 Behavior ..... 94
5B-8 Non-Electrical Energy Object ..... 96
5B-8.1 Revision History ..... 96
5B-8.2 Class Attributes ..... 96
5B-8.3 Instance Attributes ..... 97
5B-8.3.1 Semantics ..... 98
5B-8.4 Common Services ..... 102
5B-8.4.1 Get_Attributes_All Response ..... 103
5B-8.5 Behavior ..... 103
5B-9 Power Management Object ..... 106
5B-9.1 Revision History ..... 107
5B-9.2 Class Attributes ..... 108
5B-9.3 Instance Attributes ..... 108
5B-9.3.1 Semantics ..... 110
5B-9.4 Common Services ..... 113
5B-9.4.1 Get_Attributes_All Response ..... 113
5B-9.5 Object Specific Services ..... 114
5B-9.5.1 Power_Management Service ..... 114
5B-9.5.2 Set_Pass_Code Service ..... 115
5B-9.5.3 Clear_Pass_Code Service ..... 116
5B-9.6 Behavior ..... 117
5B-9.6.1 Power Management States ..... 117
5B-9.6.2 Ownership Relationship between Client and Server ..... 119
5B-9.6.3 Transactions between Client and Server ..... 120
5B-9.7 Power Management Messaging Sequences ..... 123
5B-9.8 Modbus Integration Considerations ..... 126
5B-10 Power Curtailment Object ..... 127
5B-10.1 Class Attributes ..... 128
5B-10.2 Instance Attributes ..... 129
5B-10.2.1 Semantics ..... 130
5B-10.3 Common Services ..... 134
5B-10.3.1 Get_Attributes_All Response ..... 135
5B-10.3.2 Create Service Description ..... 135
5B-10.4 Object Specific Services ..... 137
5B-10.4.1 Establish_Ownership Service ..... 137
5B-10.4.2 Release_Ownership Service ..... 139
5B-10.4.3 Change_Ownership Service ..... 140
5B-10.4.4 Go_To_Level Service ..... 141
5B-10.4.5 Go_To_Level_Query Service ..... 143
5B-10.4.6 Capture_Level Service ..... 146
5B-10.4.7 Remove_Level Service ..... 148
5B-10.4.8 Revise_Level Service ..... 149
5B-10.4.9 Read_Level Service ..... 152
5B-10.4.10 Write_Level Service ..... 153


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

5B-10.5 Behavior ..... 156
5B-10.5.1 Power Curtailment States ..... 156
5B-10.5.2 Ownership Relationship between Client and Managed Instance ..... 157
5B-10.5.3 Curtailment Level assignment ..... 157
5B-10.5.4 Transactions between Client and Managed Instance ..... 158
5B-10.5.5 Updating Status Information ..... 158
5B-10.6 Power Curtailment Messaging Sequences ..... 159
5B-10.6.1 Establish Ownership ..... 159
5B-10.6.2 Reestablish Ownership ..... 159
5B-10.6.3 Preview a Power Curtailment Command Response ..... 160
5B-10.6.4 Command to Curtailment Level ..... 160
5B-10.6.5 Obtain Exclusive Access to a Managed Instance ..... 161
5B-10.6.6 Capture a New Curtailment Level ..... 161
5B-10.6.7 Backup Curtailment Levels ..... 162
5B-10.6.8 Restore Curtailment Levels ..... 163
5B-10.6.9 Create a New Power Curtailment Object Instance ..... 163
5B-11 Connection Configuration Object ..... 164
5B-11.1 Class Attributes ..... 164
5B-11.1.1 Semantics ..... 166
5B-11.2 Instance Attributes ..... 167
5B-11.2.1 I/O Mapping Formats ..... 170
5B-11.2.2 Semantics ..... 170
5B-11.3 Connection Configuration Object Change Control ..... 174
5B-11.4 Common Services ..... 175
5B-11.4.1 Get_Attributes_All (Service Code 0x01) ..... 175
5B-11.4.2 Set_Attributes_All (Service Code 0x02) ..... 177
5B-11.4.3 Create (Service Code 0x08) ..... 179
5B-11.4.4 Delete (Service Code 0x09) ..... 180
5B-11.4.5 Restore (Service Code 0x15) ..... 180
5B-11.4.6 Object-specific Services ..... 181
5B-11.4.7 Kick_Timer (Service Code 0x4B) ..... 181
5B-11.4.8 Open_Connection (Service Code 0x4C) ..... 181
5B-11.4.9 Close_Connection (Service Code 0x4D) ..... 181
5B-11.4.10 Stop_Connection (Service Code 0x4E) ..... 181
5B-11.4.11 Change_Start (Service Code 0x4F) ..... 182
5B-11.4.12 Get_Status (Service Code 0x50) ..... 182
5B-11.4.13 Change_Complete (Service Code 0x51) ..... 183
5B-11.4.14 Audit_Changes (Service Code 0x52) ..... 183
5B-11.5 Behavior ..... 184


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-1 Object Library Continued

This is the continuation of the object library from Chapter 5A.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5B-2 Time Sync Object

**Class Code: 43 Hex**

## 5B-2.1 Time Sync Object Overview

The Time Sync Object provides a CIP interface to the IEEE 1588 (IEC 61588) Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, commonly referred to as the Precision Time Protocol (PTP). Refer to the IEEE 1588 (IEC 61588) Standard for additional details.

Any device supporting CIP Sync shall provide a single instance (Instance 1) of the Time Sync Object. CIP Sync is described in detail starting at section 5B-2.7.

The object provides attributes and services to:

1. Get clock status and properties such as synchronized state, current offset to master, and grandmaster identity.
2. Access PTP clock management functions such as clock priority.
3. Access the PTP network of devices via native PTP management messages.

## 5B-2.2 Revision History

**Table 5B-2-1 Revision History**

| Revision | History |
|---|---|
| 01 | Initial Definition - Obsolete |
| 02 | Attribute changes to support 1588 version 2.<br>Refer to section 5B-2.20 for a detailed list of attribute changes. |
| 03 | Clarified “NV-ness” of all attributes<br>Changed “NV-ness” of PTPEnable (1) attribute<br>Clarified that PTPEnable (1) attribute only applies to ordinary or boundary clocks |

## 5B-2.3 Class Attributes

The Time Sync Object shall support the following class attributes. Refer to the IEEE 1588 specification as the authoritative reference for semantic details.

**Table 5B-2-2 Time Sync Object Class Attributes**

| Attrib ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of values |
|---|---|---|---|---|---|---|
| 1 | Required | Get | Revision | UINT | Revision of Object | The current value assigned to this attribute is three (3). |
| 2 thru 7 | These class attributes are optional and are described in Chapter 4. | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-2.4 Instance Attributes

The Time Sync Object supports the following instance attributes.Refer to the 1588 PTP specification for additional details.

**Table 5B-2-3 Time Sync Object Instance Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of values |
|---|---|---|---|---|---|---|---|
| 1 | Required | Set | NV[1] | PTPEnable | BOOL | PTP Enable | See 5B-2.4.1 |
| 2 | Required | Get | V | IsSynchronized | BOOL | Local clock is synchronized with master | See 5B-2.4.2 |
| 3 | Required | Set [3] | V | SystemTimeMicroseconds | ULINT | Current value of system_time in microseconds | See 5B-2.4.3 |
| 4 | Required | Set [3] | V | SystemTimeNanoseconds | ULINT | Current value of system_time in nanoseconds | See 5B-2.4.4 |
| 5 | Required | Get | V | OffsetFromMaster | LINT | Offset between local clock and master clock | See 5B-2.4.5 |
| 6 | Required | Set | V | MaxOffsetFromMaster | ULINT | Maximum offset between local clock and master clock | See 5B-2.4.6 |
| 7 | Required | Get | V | MeanPathDelayToMaster | LINT | Mean path delay to master | See 5B-2.4.7 |
| 8 | Required | Get | V | GrandMasterClockInfo | STRUCT of | Grandmaster Clock Info | See 5B-2.4.8 |
| | | | | ClockIdentity | USINT[8] | Clock Identity | |
| | | | | ClockClass | UINT | Clock class | |
| | | | | TimeAccuracy | UINT | Time Accuracy | |
| | | | | OffsetScaledLogVariance | UINT | Offset scaled log variance | |
| | | | | CurrentUtcOffset | UINT | Current UTC offset | |
| | | | | TimePropertyFlags | WORD | Time property flags | |
| | | | | TimeSource | UINT | Time source | |
| | | | | Priority1 | UINT | Priority1 | |
| | | | | Priority2 | UINT | Priority2 | |
| 9 | Required | Get | V | ParentClockInfo | STRUCT of | Parent Clock Info | See 5B-2.4.9 |
| | | | | ClockIdentity | USINT[8] | Clock Identity | |
| | | | | PortNumber | UINT | Port Number | |
| | | | | ObservedOffsetScaledLogVariance | UINT | Observed offset scaled log variance | |
| | | | | ObservedPhaseChangeRate | UDINT | Observed phase change rate | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Time Sync Object, Class Code: $43_{hex}$

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of values |
|---|---|---|---|---|---|---|---|
| 10 | Required | Get | V | LocalClockInfo | STRUCT of | Local Clock Info | See 5B-2.4.10 |
| | | | | ClockIdentity | USINT[8] | Clock identity | |
| | | | | ClockClass | UINT | Clock Class | |
| | | | | TimeAccuracy | UINT | Time accuracy | |
| | | | | OffsetScaledLogVariance | UINT | Offset scaled log variance | |
| | | | | CurrentUtcOffset | UINT | Current UTC offset | |
| | | | | TimePropertyFlags | WORD | Time property flags | |
| | | | | TimeSource | UINT | Time source | |
| 11 | Required | Get | V | NumberOfPorts | UINT | Number of ports | See 5B-2.4.11 |
| 12 | Required | Get | V | PortStateInfo | STRUCT of | Port state info | See 5B-2.4.12 |
| | | | | NumberOfPorts | UINT | Number of ports | |
| | | | | | ARRAY of STRUCT | | |
| | | | | PortNumber | UINT | Port number | |
| | | | | PortState | UINT | Port state | |
| 13 | Required | Set | NV | PortEnableCfg | STRUCT of | Port enable cfg | See 5B-2.4.13 |
| | | | | NumberOfPorts | UINT | Number of ports | |
| | | | | | ARRAY of STRUCT | | |
| | | | | PortNumber | UINT | Port number | |
| | | | | PortEnable | UINT | Port enable | |
| 14 | Required | Set | NV | PortLogAnnounceIntervalCfg | STRUCT of | Port log announce interval cfg | See 5B-2.4.14 |
| | | | | NumberOfPorts | UINT | Number of ports | |
| | | | | | ARRAY of STRUCT | Port log announce interval of each port | |
| | | | | PortNumber | UINT | Port number | |
| | | | | PortLogAnnounceInterval | UINT | Port log announce interval | |
| 15 | Required | Set | NV | PortLogSyncIntervalCfg | STRUCT of | Port log sync interval cfg | See 5B-2.4.15 |
| | | | | NumberOfPorts | UINT | Number of ports | |
| | | | | | ARRAY of STRUCT | | |
| | | | | PortNumber | UINT | Port number | |
| | | | | PortLogSyncInterval | INT | Port log sync interval | |
| 16 | Conditional [2] | Set | NV | Priority1 | USINT | Priority1 | See 5B-2.4.16 |
| 17 | Conditional [2] | Set | NV | Priority2 | USINT | Priority2 | See 5B-2.4.17 |
| 18 | Required | Set | NV | DomainNumber | USINT | Domain number | See 5B-2.4.18 |
| 19 | Required | Get | V | ClockType | WORD | Clock type | See 5B-2.4.19 |
| 20 | Required | Get | V | ManufactureIdentity | USINT[4] | Manufacture identity | See 5B-2.4.20 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Time Sync Object, Class Code: $43_{hex}$

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of values |
|---|---|---|---|---|---|---|---|
| 21 | Required | Get | V | ProductDescription | STRUCT of | Product description | See 5B-2.4.21 |
| | | | | Size | UDINT | Size of product description | |
| | | | | Description | ARRAY of USINT | Description | |
| 22 | Required | Get | V | RevisionData | STRUCT of | Revision data | See 5B-2.4.22 |
| | | | | Size | UDINT | Size of revision data | |
| | | | | Revision | ARRAY of USINT | Revision | |
| 23 | Required | Get | V | UserDescription | STRUCT of | User description | See 5B-2.4.23 |
| | | | | Size | UDINT | Size of user description | |
| | | | | Description | ARRAY of USINT | Description | |
| 24 | Required | Get | V | PortProfileIdentityInfo | STRUCT of | Port profile identity info | See 5B-2.4.24 |
| | | | | NumberOfPorts | UINT | Number of ports | |
| | | | | | ARRAY of STRUCT | | |
| | | | | PortNumber | UINT | Port number | |
| | | | | PortProfileIdentity | USINT[8] | Port profile identity | |
| 25 | Required | Get | V | PortPhysicalAddressInfo | STRUCT of | Port physical address info | See 5B-2.4.25 |
| | | | | NumberOfPorts | UINT | Number of ports | |
| | | | | | ARRAY of STRUCT | | |
| | | | | PortNumber | UINT | Port number | |
| | | | | PhysicalProtocol | USINT[16] | Physical protocol | |
| | | | | SizeOfAddress | UINT | Size of address | |
| | | | | PortPhysicalAddress | USINT[16] | Port physical address | |
| 26 | Required | Get | V | PortProtocolAddressInfo | STRUCT of | Port protocol address info | See 5B-2.4.26 |
| | | | | NumberOfPorts | UINT | Number of ports | |
| | | | | | ARRAY of STRUCT | | |
| | | | | PortNumber | UINT | Port number | |
| | | | | NetworkProtocol | UINT | Network protocol | |
| | | | | SizeOfAddress | UINT | Size of address | |
| | | | | PortProtocolAddress | USINT[16] | Port protocol address | |
| 27 | Required | Get | V | StepsRemoved | UINT | Steps removed | See 5B-2.4.27 |
| 28 | Required | Get | V | SystemTimeAndOffset | STRUCT of | System time and offset | See 5B-2.4.28 |
| | | | | SystemTime | ULINT | System time | |
| | | | | SystemOffset | ULINT | System offset | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Time Sync Object, Class Code: $43_{hex}$**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of values |
|---|---|---|---|---|---|---|---|

Table Footnotes:

1 Modular devices with a boundary clock may treat PTPEnable as volatile if another module provides the value

2 Required for master capable devices

3 Set is required for master clock capable devices and is optional otherwise.

### 5B-2.4.1 Attribute 1: PTPEnable

PTPEnable specifies the enable status for the Precision Time Protocol on the device.

PTPEnable does not apply to the transparent clock function(s) within a device. If the transparent clock functions of a device are configurable, they are managed through vendor specific means.

**Table 5B-2-4 PTPEnable Attribute Values**

| PTP Enable | Value |
|---|---|
| Enabled | 1 |
| Disabled | 0 |

The default values prescribed in Table 5B-2.5 were chosen to simplify initial system construction for most cases. There are conflicting requirements for devices implementing a boundary clock. For instance, if the device is designed to serve the role of a CIP Sync master (e.g. a controller with multiple PTP ports), then it should default to Disabled. However, when possible, boundary clocks should default to Enabled.

**Table 5B-2-5 PTPEnable Attribute Default Values**

| PTP Clock Type | Default Value |
|---|---|
| Slave-only Ordinary Clock | Enabled |
| Boundary Clock | Product specific, Enabled preferred |
| Master-capable Ordinary Clock | Disabled |
| Transparent Clock | Not applicable |

### 5B-2.4.2 Attribute 2: IsSynchronized

IsSynchronized specifies whether the local clock is synchronized with a master reference clock. The value is 1 if the local clock is synchronized and 0 if the local clock is not synchronized. At a minimum, a clock is synchronized if it has one port in the slave state and is receiving updates from the time master. A particular device may impose additional criteria or constraints on synchronization which are outside the scope of this specification . For example, a device may specify a sync threshold of 10 microseconds and only indicate that the device is synchronized when the OffsetFromMaster (attribute 5) is less than 10 microseconds for a period of 5 sync intervals (attribute 15).

**Table 5B-2-6 IsSynchronized Attribute Values**

| Synchronization Status | Value |
|---|---|
| Synchronized | 1 |
| Not synchronized | 0 |

### 5B-2.4.3 Attribute 3: SystemTimeMicroseconds

SystemTimeMicroseconds specifies a 64-bit value of the current system time in units of microseconds. See 5B-2.10.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-2.4.4 Attribute 4: SystemTimeNanoseconds

SystemTimeNanoseconds specifies a 64-bit value of the current system time in units of nanoseconds. See 5B-2.10.

### 5B-2.4.5 Attribute 5: OffsetFromMaster

OffsetFromMaster specifies the amount of deviation between the local clock and its master clock in nanoseconds.

### 5B-2.4.6 Attribute 6: MaxOffsetFromMaster

MaxOffsetFromMaster specifies the absolute value of the maximum amount of deviation between the local clock and the master clock in nanoseconds since last set. The MaxOffsetFromMaster is settable, typically to 0.

### 5B-2.4.7 Attribute 7: MeanPathDelayToMaster

MeanPathDelayToMaster specifies the average path delay between the local clock and master clock in nanoseconds.

### 5B-2.4.8 Attribute 8: GrandmasterInfo

See attribute 10, LocalClockInfo -- members of the structure have the same semantics as attribute 10.

### 5B-2.4.9 Attribute 9: ParentClockInfo

See attribute 10, LocalClockInfo -- members of the structure have the same semantics as attribute 10.

### 5B-2.4.10 Attribute 10: LocalClockInfo

GrandmasterClockInfo, ParentClockInfo, and LocalClockInfo specify clock property information for the Grandmaster, Parent and Local PTP clock respectively. The data is extracted from the PTP data sets maintained by the PTP device.

The Clock Identity provides a unique identifier for the clock. The clock time source, class, variance, and other attributes provide additional information about the properties of the clock.

#### 5B-2.4.10.1 ClockIdentity

ClockIdentity specifies the unique identifier for the clock. The format of the identifier depends on the network protocol. Ethernet encodes the MAC address into the identifier. DeviceNet and ControlNet encode the Vendor ID and Serial Number.

**Table 5B-2-7 ClockIdentity Encoding for Different Network Implementations**

| Network | Octet 0 | Octet 1 | Octet 2 | Octet 3 | Octet 4 | Octet 5 | Octet 6 | Octet 7 |
|---|---|---|---|---|---|---|---|---|
| EtherNet/IP | MAC Address[1] | | | FF | FE | MAC Address[1] | | |
| DeviceNet | FF | 01 | Vendor ID[2] | | Serial Number[2] | | | |
| ControlNet | FF | 02 | Vendor ID[2] | | Serial Number[2] | | | |
| Local or closed | FF | FF | Vendor ID[2] | | Serial Number[2] | | | |
| All others | see 1588-2008 PTP specification | | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Table Footnotes:

1. Most significant octet of MAC Address is assigned to most significant octet of ClockIdentity. For example, a MAC address of 00 01 02 03 04 05, results in a clock identity of 00 01 02 FF FE 03 04 05
2. The Vendor ID and Serial Number are formatted as big endian in the ClockIdentity. For example, a Vendor ID of 0809 and a Serial Number of 03040506 results in a clock identity for DeviceNet of FF 01 08 09 03 04 05 06

### 5B-2.4.10.2 PortNumber

PortNumber specifies the port number of the port identity.

### 5B-2.4.10.3 ClockClass

ClockClass specifies the class of the clock quality. The clock class represents a relative measure of the clock quality used by the Best Master algorithm to determine the grandmaster. The class is a value between 0 and 255, with 0 as the best clock. These are the values most likely to be used in CIP Sync. Refer to the 1588 specification for a complete list of values.

**Table 5B-2-8 Clock Class Values**

| Clock Class | Value |
|---|---|
| Primary Reference | 6 |
| Primary Reference (Hold) | 7 |
| Degraded Reference A (Master only) | 52 |
| Degraded Reference B (Master / Slave) | 187 |
| Default | 248 |
| Slave Only | 255 |
| see 1588 specification | all other |

### 5B-2.4.10.4 TimeAccuracy

TimeAccuracy specifies the expected absolute accuracy of the clock relative to the PTP epoch. TimeAccuracy is the accuracy measure of clock quality used by the Best Master algorithm to determine the grandmaster. The accuracy is specified as a graduated scale starting at 25 nsec and ending at greater than 10 seconds or unknown. A GPS time source will have an accuracy of approximately 250 nanoseconds. A HAND set clock will typically have accuracy less than 10 seconds. The lower the accuracy value, the better the clock.

**Table 5B-2-9 TimeAccuracy Values**

| Clock Accuracy | Value |
|---|---|
| Reserved | 0x00-0x1F |
| The time is accurate to within 25 ns | 0x20 |
| The time is accurate to within 100 ns | 0x21 |
| The time is accurate to within 250 ns | 0x22 |
| The time is accurate to within 1 us | 0x23 |
| The time is accurate to within 2.5 us | 0x24 |
| The time is accurate to within 10 us | 0x25 |
| The time is accurate to within 25 us | 0x26 |
| The time is accurate to within 100 us | 0x27 |
| The time is accurate to within 250 us | 0x28 |
| The time is accurate to within 1 ms | 0x29 |
| The time is accurate to within 2.5 ms | 0x2A |
| The time is accurate to within 10 ms | 0x2B |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Clock Accuracy | Value |
|---|---|
| The time is accurate to within 25 ms | 0x2C |
| The time is accurate to within 100 ms | 0x2D |
| The time is accurate to within 250 ms | 0x2E |
| The time is accurate to within 1 s | 0x2F |
| The time is accurate to within 10 s | 0x30 |
| The time is accurate to >10 s | 0x31 |
| reserved | 0x32-0x7F |
| For use by alternate PTP profiles | 0x80-0xFD |
| Unknown | 0xFE |
| reserved | 0xFF |

### 5B-2.4.10.5 OffsetScaledLogVariance

OffsetScaledLogVariance specifies a measure of the inherent stability properties of the clock. OffsetScaledLogVariance is the variance measure of clock quality used by the Best Master algorithm to determine the grandmaster. The value is represented in offset scaled log units. The lower the variance, the better the clock

### 5B-2.4.10.6 CurrentUtcOffset

CurrentUtcOffset specifies the current UTC offset in seconds from International Atomic Time (TAI) of the clock. As of 0 hours 1 January 2006 UTC, the offset was 33 seconds.

### 5B-2.4.10.7 TimePropertyFlags

TimePropertyFlags specifies the time property flags of the clock.

**Table 5B-2-10 TimePropertyFlags Bit Values**

| Time Property Flags | Bit Index |
|---|---|
| Leap indicator 61 | 0 |
| Leap indicator 59 | 1 |
| Current UTC offset valid | 2 |
| PTP timescale | 3 |
| Time traceable | 4 |
| Frequency traceable | 5 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5B-2.4.10.8 TimeSource

TimeSource specifies the primary time source of the clock.

**Table 5B-2-11 TimeSource Values**

| Time Source | Value |
|---|---|
| ATOMIC CLOCK | 0x10 |
| GPS | 0x20 |
| TERRESTRIAL RADIO | 0x30 |
| PTP | 0x40 |
| NTP | 0x50 |
| HAND SET | 0x60 |
| OTHER | 0x90 |
| INTERNAL OSCILLATOR | 0xA0 |
| For use by alternate PTP profiles | 0xF0-0xFE |
| reserved | 0xFF |

#### 5B-2.4.10.9 Priority1 and Priority2

Priority1 and Priority2 values specify the relative priority of the grandmaster clock to other clocks in the system. See priority1 and priority2 attributes 16 and 17 respectively.

#### 5B-2.4.10.10 ObservedOffsetScaledLogVariance

ObservedOffsetScaledLogVariance specifies an estimated measure of the parent clock's variance as observed by the slave clock.

#### 5B-2.4.10.11 ObservedPhaseChangeRate

ObservedPhaseChangeRate specifies an estimated measure of the parent clock's drift as observed by the slave clock.

### 5B-2.4.11 Attribute 11: NumberOfPorts

NumberOfPorts specifies the number of PTP ports on the device. PTP Ordinary clocks have one port. PTP Boundary and Transparent clocks have more than one port. A hybrid clock that contains both an ordinary clock and an end-to-end transparent clock has a value of one (1) for this attribute.

### 5B-2.4.12 Attribute 12: Port State

PortStateInfo specifies the current state of each PTP port on the device.

**Table 5B-2-12 Port State Values**

| Port State | Value |
|---|---|
| INITIALIZING | 1 |
| FAULTY | 2 |
| DISABLED | 3 |
| LISTENING | 4 |
| PRE_MASTER | 5 |
| MASTER | 6 |
| PASSIVE | 7 |
| UNCALIBRATED | 8 |
| SLAVE | 9 |
| reserved | all other |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-2.4.13 Attribute 13: PortEnableCfg

PortEnableCfg structure specifies the port enable configuration of each port on the device. The PortEnable member is set to 1 if the port is enabled and 0 if the port is disabled. The default value is Enabled.

**Table 5B-2-13 PortEnable Values**

| Port Enable | Value |
|---|---|
| Enabled | 1 |
| Disabled | 0 |

### 5B-2.4.14 Attribute 14: PortLogAnnounceIntervalCfg

PortLogAnnounceIntervalCfg structure specifies the PTP announce interval between successive “Announce” messages issued by a master clock on each PTP port of the device. The units of the PortLogAnnounceInterval member are log base 2 seconds. Refer to section 5B-2.15.2 for default values.

### 5B-2.4.15 Attribute 15: PortLogSyncIntervalCfg

PortLogSyncIntervalCfg structure specifies the PTP sync interval between successive “Sync” messages issued by a master clock on each PTP port of the device. The units of the PortLogSyncInterval member are log base 2 seconds. Refer to section 5B-2.15.2 for default values.

### 5B-2.4.16 Attribute 16: Priority1

Priority1 specifies the priority1 setting of the PTP Best Master Algorithm. This attribute specifies the Best Master ranking of this clock and superceeds the clock quality (class, accuracy, and variance). This attribute allows the user to override the automatic selection of the best master clock before any quality measures are evaluated. The value is between 0 and 255. The highest priority is 0. Refer to section 5B-2.15.2 for default values.

### 5B-2.4.17 Attribute 17: Priority2

Priority2 specifies the priority2 setting of the PTP Best Master Algorithm. This attribute specifies the Best Master ranking of this clock after clock quality (class, accuracy, and variance) has been evaluated and superceeds the tie-breaker. This attribute allows the user to override the automatic selection of the best master clock after quality measures have been evaluated – i.e. choose the best master from a set of clocks of equal quality. The value is between 0 and 255. The highest priority is 0. Refer to section 5B-2.15.2 for default values.

### 5B-2.4.18 Attribute 18: DomainNumber

DomainNumber specifies the PTP clock domain. Refer to section 5B-2.15.2 for default values.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-2.4.19 Attribute 19: ClockType

The value of ClockType shall indicate the PTP functions that the node supports as defined in Table 5B-2.14. A value of one (1) for the bit indicates that the capability applies to this node. NOTE—More than one bit may be set - for example an ordinary clock combined with an end-to-end transparent clock. When bit 8 is set, bit 7 shall also be set, as Slave Only functionality is a subset of the Ordinary Clock function.

**Table 5B-2-14 Types of Clock**

| Bit | | | | | | | |
|---|---|---|---|---|---|---|---|
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Ordinary Clock | Boundary Clock | 0 | End-to-End Transparent Clock | Management Node | 0 | 0 | 0 |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | Slave Only |

### 5B-2.4.20 Attribute 20: ManufactureIdentity

ManufactureIdentity specifies the manufacture identity of the clock. The first 3 octets specify the IEEE OUI (Organization Unigue Id) for the manufacturer. The last octet is reserved.

### 5B-2.4.21 Attribute 21: ProductDescription

ProductDescription specifies the product description of the device that contains the clock. The format is:

- The name of manufacture of the device followed by a semicolon.
- The model number of the device followed by a semicolon.
- The serial number.

For example: ACME Manufacturing;C2901;123456

The format is UTF-8 Unicode. The maximum number of symbols is 64. The Size field of the data type is the total number of bytes for the Description field. Convert the number of symbols to bytes.

### 5B-2.4.22 Attribute 22: RevisionData

RevisionData specifies the revision data of the device that contains the clock. The format is:

- The hardware revision of the clock followed by a semicolon.
- The firmware revision of the clock followed by a semicolon.
- The software revision of the clock.

For example: 1.2;2.3;3.0.1

The format is UTF-8 Unicode. The maximum number of symbols is 32. The Size field of the data type is the total number of bytes for the Revision field. Convert the number of symbols to bytes. Subfields that don’t apply may be null or blank (e.g. “1.2;2.3;” or “;;3.4”).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-2.4.23 Attribute 23: UserDescription

UserDescription specifies the user description of the device that contains the clock. The format is:

- A user defined name or description of the device followed by a semicolon.
- A user defined physical location of the device.

For example: Sensor-1; Section-7 Cabinet-2 Rack-3

The format is UTF-8 Unicode. The maximum number of symbols is 128. The Size field of the data type is the total number of bytes for the Description Field. Convert the number of symbols to bytes.

### 5B-2.4.24 Attribute 24: PortProfileIdentityInfo

PortProfileIdentityInfo specifies the PTP profile of each port of the device. The attribute returns the profile identity of the currently active profile. The profile identity is contained in the first 6 bytes of the array. The last two octets should be set to zero. Refer to section 5B-2.15.1.

### 5B-2.4.25 Attribute 25: PortPhysicalAddressInfo

The PortPhysicalAddressInfo specifies the Physical Protocol and Physical Address of each port of the device.

The Physical Protocol is an array of ASCII characters and is limited to a maximum of 16 characters. Unused array characters are zero-filled. See Table 5B-2.15 for a list of recommended values. For example a physical protocol of “IEEE 802.3” = 49 45 45 45 20 38 30 32 2E 33 00 00 00 00 00 00.

The Physical Address is an array of bytes and is limited to a maximum of 16 bytes. Unused array bytes are zero-filled. For example a Physical Address for MAC address 01 02 03 04 05 06 = 01 02 03 04 05 06 00 00 00 00 00 00 00 00 00 00 and has a size of 6 bytes.

**Table 5B-2-15 Network Protocol to Physical Protocol Mapping**

| Network Protocol | Physical Protocol | Physical Address |
|---|---|---|
| UDP/IPv4 (e.g. EtherNet/IP) | “IEEE 802.3” | Use MAC address |
| UDP/IPv6 | “IEEE 802.3” | Use MAC address |
| IEEE 802.3 | “IEEE 802.3” | Use MAC address |
| DeviceNet | “DeviceNet” | Use MAC ID |
| ControlNet | “ControlNet” | Use MAC ID |
| Local or unknown protocol | Vendor specific | Use vendor specific |

### 5B-2.4.26 Attribute 26: PortProtocolAddressInfo

The PortProtocolAddressInfo specifies the network protocol and protocol address of each port of the device (e.g. IP address). The Network Protocol specifies the protocol for the network. See Table 5B-2.16. The maximum number of bytes for the PortProtocolAddress is 16. Unused array bytes are zero-filled. For example a Network Protocol of UDP/IPv4 = 0001 and an IP protocol address of 192.168.1.2 = C0 A8 01 02 00 00 00 00 00 00 00 00 00 00 00 00 and has a size of 4 bytes.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-2-16 Network Protocol Values**

| Network Protocol | Value |
|---|---|
| UDP/IPv4 (e.g. EtherNet/IP) | 1 |
| UDP/IPv6 | 2 |
| IEEE 802.3 | 3 |
| DeviceNet | 4 |
| ControlNet | 5 |
| Local or unknown protocol | 0xFFFE |
| reserved | all other |

### 5B-2.4.27 Attribute 27: StepsRemoved

StepsRemoved specifies the number of communication paths traversed between the local clock and the grandmaster clock.

### 5B-2.4.28 Attribute 28: SystemTimeAndOffset

SystemTimeAndOffset specifies the System Time in microseconds and the Offset to the local clock value. The responding device will return the current System Time and Offset. Refer to section 5B-2.16, CIP Sync Clock Model.

## 5B-2.5 Common Services

The Time Sync Object provides the following common services.

**Table 5B-2-17 Common services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 01 hex | Optional | N/A | Get_Attribures_All | See Appendix A |
| 03 hex | N/A | Required | Get_Attributes_List | See Appendix A |
| 04 hex | N/A | Required | Set_Attributes_List | See Appendix A |
| 0E hex | Required | Required | Get_Attribute_Single | See Appendix A |
| 10 hex | N/A | Required | Set_Attribute_Single | See Appendix A |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-2.5.1 Get_Attributes_All Response

#### 5B-2.5.1.1 Class Level

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5B-2-18 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | |
| 2 | UINT | Max Instance | 0 |
| 3 | UINT | Number of Instances | 0 |
| 4 | STRUCT of: | Optional Attribute List | |
| | UINT | Number of attributes | 0 |
| | ARRAY of UINT | Optional attributes | (null) |
| 5 | STRUCT of: | Optional Service List | |
| | UINT | Number of services | 0 |
| | ARRAY of UINT | Optional services | (null) |
| 6 | UINT | Maximum ID Number Class Attributes | 0 |
| 7 | UINT | Maximum ID Number Instance Attributes | 0 |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

## 5B-2.6 Object-Specific Services

The Time Sync Object does not define any object-specific services.

## 5B-2.7 CIP Sync Overview

CIP Sync is the name given to the time synchronization technology for the Common Industrial Protocol (CIP). This technology will allow accurate real-time synchronization of devices and controllers connected over CIP networks that require time stamping, sequence of events recording, distributed motion control, and support for highly distributed applications requiring increased control coordination.

CIP Sync is based on the IEEE 1588 (IEC 61588) standard – Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, commonly referred to as the Precision Time Protocol (PTP). The protocol is designed for, but not limited to, local area networks like Ethernet. The protocol provides a standard mechanism to synchronize clocks across a network of distributed devices. A copy of the standard may be obtained by ordering the standard IEEE 1588-2008 from the IEEE Standards Organization Committee.

The Time Sync Object provides a CIP interface to the IEEE 1588 (IEC 61588) Standard.

## 5B-2.8 Definitions

**Best Master Clock Algorithm (BMCA):** The algorithm performed by each node to determine the clock that will become the master clock on a subnet and the grandmaster clock for the domain. The algorithm primarily compares priority1, clock quality, priority2, and source identity to determine the best master among available candidates.

**Boundary clock:** A clock that has multiple PTP ports in a domain and maintains the timescale used in the domain. It may serve as the source of time, i.e.: be a master clock, and may synchronize to another clock, i.e.: be a slave clock.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Clock:** A node participating in the PTP protocol that is capable of providing a measurement of the passage of time since a defined epoch. There are three types of clocks in 1588: boundary, transparent and ordinary clocks.

**Domain:** A logical grouping of clocks that synchronize to each other using the PTP protocol, but that are not necessarily synchronized to clocks in another domain.

**Epoch:** The origin of a time scale. For example, Jan 1, 1970.

**Grandmaster clock:** Within a domain, a clock that is the ultimate source of time for clock synchronization using the PTP protocol.

**Master clock:** In the context of a single PTP communication path, a clock that is the source of time to which all other clocks on that path synchronize.

**Ordinary clock:** A clock that has a single PTP port in a domain and maintains the timescale used in the domain. It may serve as a source of time, i.e. be a master clock, or may synchronize to another clock, i.e. be a slave clock.

**Parent clock:** The master clock to which a clock is synchronized.

**Precision Time Protocol (PTP)**: The protocol defined by the 1588 standard.

**PTP message:** One of the message types defined by the 1588 standard.

**PTP port:** A logical access point of a clock for PTP communications to the communications network.

**Synchronized clocks:** Two clocks are synchronized to a specified uncertainty if they have the same epoch and their measurements of the time of a single event at an arbitrary time differ by no more than that uncertainty.

**System Time:** The absolute time value as defined by CIP Sync in the context of a distributed time system where all devices have a local clock that is synchronized with a common master clock. System Time is a 64-bit integer value in units of nanoseconds or microseconds with a value of 0 corresponding to an epoch of January 1, 1970.

**Transparent Clock:** A device that measures the time taken for a PTP event message to transit the device and provides this information to clocks receiving this PTP event message.

## 5B-2.9 Overview of the IEEE 1588 Precision Time Protocol (PTP)

The IEEE 1588 standard specifies a protocol to synchronize independent clocks running on separate nodes of a distributed measurement and control system to a high degree of accuracy and precision. The clocks communicate with each other over a communication network. In its basic form, the protocol is intended to be administration free. The protocol generates a master slave relationship among the clocks in the system. Within a given subnet of a network there will be a single master clock. All clocks ultimately derive their time from a clock known as the grandmaster clock.

A PTP system of distributed clocks consists of ordinary, boundary, and transparent clocks. An ordinary clock may be a time master or a time slave. A boundary clock is a clock with more than one port and separates two or more distinct communication paths. A transparent clock also has more than one port, but differs from a boundary clock by transparently passing PTP messages between its ports. Typically boundary and transparent clocks are implemented in switches.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

One clock on each subnet in the system is selected as the master clock. The selection of a master is made by each of the other clocks by examining information contained in PTP Announce messages. An announce message is sent periodically by any port claiming to be the master clock. All ports use the same algorithm, termed the Best Master Clock Algorithm (BMCA). If a port of a master clock receives an Announce message from a better clock then that port will cease to be a master port and will become a slave port. Likewise if a clock with a slave port determines that it would make a better master than the current master clock or if there is no current master clock, the port becomes a master port and begins to send Announce messages. Some nodes may be implemented as slave only nodes and never assume mastership (e.g. an I/O device). Other nodes may be implemented as master only nodes and never become slaves (e.g. primary clock source). The master and slaves issue timing messages: Sync, Followup, Delay Request, and Response to facilitate the synchronization of clocks between the master and slave.

An example configuration is pictured in Figure 5B-2.1. A system will typically consist of one or more devices capable of becoming a master clock (i.e. clock sources, controllers), slave only clocks (i.e. I/O), and infrastructure devices that may or may not support time synchronization (boundary, transparent, and non-1588 switches, etc.). The system may also contain hybrid devices that contain, for example, both an ordinary clock and a transparent clock.

**Figure 5B-2.1 Example System with Grandmaster, Boundary, and Slave Clocks**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-2.10 CIP Sync System Time Definition

The starting date/time for CIP Sync System Time is 12:00 a.m., January 1, 1970. System Time is represented as a 64-bit value (ULINT) in microseconds (attribute 3) or nanoseconds (attribute 4). System Time is adjusted for leap seconds, but not local time zones or daylight savings time. It represents the current time at the 0th meridian.

In the Precision Time Protocol, time is distributed as the number of nanoseconds since the beginning of the PTP epoch, 31 December, 1969 23:59:51.999918 UTC. Leap seconds are distributed by the Precision Time Protocol as the "current UTC offset". As of 0 hours, 1 January 2006 UTC, the number of leap seconds was 33.

PTP time and leap seconds are converted to microseconds (attribute 3) or nanoseconds (attribute 4) to give System Time as:

SystemTime = PTPTime - currentUTCoffset"

## 5B-2.11 CIP Sync Implementation

Each node on the CIP network interested in clock synchronization will contain an implementation of a PTP clock as well as an implementation of the PTP software protocol stack.

### 5B-2.11.1 PTP Clock Implementation

A PTP clock can be implemented using a hardware assist circuit or with software using a free running timer. The choice of whether to implement a hardware assist clock or a software clock will depend on cost and accuracy considerations. The hardware assist circuit will generally result in a more accurate clock usually in the sub-microsecond range while a software implemented clock will yield accuracy in the tens of microseconds range. The 1588 specification does not describe the clock implementation, but does contain guidelines and requirements.

### 5B-2.11.2 PTP Protocol Stack Implementation

The PTP protocol stack implementation contains the software that implements the 1588 PTP protocol. This includes the messaging between clocks to synchronize slave clocks to master clocks, the algorithms to process message timestamps and tune the clock, and the management message support required for clock administration. The implementation of the basic protocol is straightforward.

## 5B-2.12 1588 PTP Version support

CIP Sync devices shall support version 2 of the 1588 PTP protocol.

## 5B-2.13 Clock Identity

Each PTP clock shall assign a unique 8-byte identifier for the clock. The assignment of identifiers depends on the CIP network. For Ethernet IP the identifiers are based on the MAC address. For other CIP networks, the identifiers are based on the Vendor Id and Serial Number. A local bus may also implement the PTP protocol and assign the identifier using the encoding for a local or closed network.

The identifiers are formatted as specified in Table 5B-2.19.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-2-19 Clock Identity**

| Network | Octet 0 | Octet 1 | Octet 2 | Octet 3 | Octet 4 | Octet 5 | Octet 6 | Octet 7 |
|---|---|---|---|---|---|---|---|---|
| EtherNet/IP | MAC Address[1] | | | FF | FE | MAC Address[1] | | |
| DeviceNet | FF | 01 | Vendor ID[2] | | Serial Number[2] | | | |
| ControlNet | FF | 02 | Vendor ID[2] | | Serial Number[2] | | | |
| Local or closed | FF | FF | Vendor ID[2] | | Serial Number[2] | | | |
| All others | see 1588-2008 PTP specification | | | | | | | |

Table Footnotes:

1. Most significant octet of MAC Address is assigned to most significant octet of ClockIdentity. For example, a MAC address of 00 01 02 03 04 05, results in a clock identity of 00 01 02 FF FE 03 04 05
2. The Vendor ID and Serial Number are formatted as big endian in the ClockIdentity. For example, a Vendor ID of 0809 and a Serial Number of 03040506 results in a clock identity for DeviceNet of FF 01 08 09 03 04 05 06

## 5B-2.14 PTP Port Assignments

Each device network port is given a unique PTP port number per the 1588 specification. In order to be consistent with the CIP protocols, the PTP and CIP port numbers should be the same where possible.

## 5B-2.15 CIP Sync PTP Profile

As defined by the 1588 specification, the purpose of a PTP profile is to allow organizations to specify unique selections of attribute values and optional features of the protocol that, when using the same transport protocol, will inter-work and achieve a performance that meets the requirements of a particular application.

This section defines the PTP profile for the CIP networks. It is based on the Delay Request-Response Default PTP profile defined in the 1588 specification. Note: The identification and default settings of the CIP Sync profile will be the same as the 1588 PTP default specification.

### 5B-2.15.1 Profile Identification

Table 5B-2.20 identifies the CIP PTP profile.

**Table 5B-2-20 Profile Identification**

| Profile Identifier | Specification |
|---|---|
| PTP profile description | CIP Sync Profile |
| Version | 1.0 |
| Profile identifier | 00-21-6C-00-01-00 |
| Specifying organization | ODVA |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-2.15.2 Profile Default Settings and Ranges

The CIP Sync profile shall support the PTP default and range settings as defined in Table 5B-2.21. The table defines the required range settings for this profile, but a device may support a larger range (e.g.: Log sync interval -3 to +2) as long as those values fall within the range of the IEEE 1588-2008 specification acceptable values.

**Table 5B-2-21 Profile Default Settings and Ranges**

| PTP Attribute | Attribute ID | Default[1] | Required Range | Acceptable Range | Units |
|---|---|---|---|---|---|
| Log announce interval | 14 | 1 | 0 to 4 | 0 to 4 | $Log_2$ seconds |
| Log sync interval | 15 | 0 | -1 to +1 | -3 to +2 | $Log_2$ seconds |
| Priority1 | 16 | 128 | 0 to 255 | 0 to 255 | |
| Priority2 | 17 | 128 | 0 to 255 | 0 to 255 | |
| Domain | 18 | 0 | 0 to 127 | 0 to 127 | |
| Announce receipt timeout interval | | 3 | 3 | 2 to 10 | Announce Intervals |

Table Footnotes:

1. The profile default settings may differ for ports connected to a proprietary backplane or bus.

### 5B-2.15.3 Profile Transports

Table 5B-2.22 identifies the IEEE 1588-2008 transport specified for each CIP network and a corresponding reference in the 1588 specification.

**Table 5B-2-22 Profile Transports**

| Network | IEEE 1588-2008 Transport | IEEE 1588-2008 Reference |
|---|---|---|
| EtherNet/IP | Transport of PTP over UDP/IPv4 | Annex D |
| DeviceNet | Transport of PTP over DeviceNet | Annex G |
| ControlNet | Transport of PTP over ControlNet | Annex H |

### 5B-2.15.4 PTP Domain

A CIP network shall be limited to one PTP domain. The 1588 specification allows for the presence of more than one clock domain on a network. However, for some implementations of PTP, based on 1588 version 1 hardware, the presence of other clock domains will cause interference. Therefore, to avoid interference problems a CIP network will be limited to one domain. It is recommended that new hardware implementations be designed to tolerate the presence of multiple domains.

### 5B-2.15.5 PTP Node management

Each device implementing CIP Sync shall implement an instance of the Time Sync Object. In addition the device shall implement the management message mechanism defined by the 1588 specification.

### 5B-2.15.6 Transparent Clock

Devices with multiple ingress/egress ports shall implement an end-to-end transparent clock between those ports.

### 5B-2.15.7 Path delay mechanism

Each device implementing CIP Sync shall implement the delay request-response path delay measurement mechanism.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-2.15.8 Recommended PTP Clock Settings

This section defines recommended default settings for PTP clock attributes. Refer to Table 5B-2.23. These settings provide general recommendations for various types of devices to assign a relative priority used by the Best Master Clock Algorithm (BMCA) of the PTP protocol. Typically controllers, switches, sources will become masters over I/O because they have a better settings.

Grandmaster capable devices may dynamically adjust clock class, clock accuracy, and time source settings. For example, a GPS clock will dynamically change its clock class and accuracy once it has locked onto satellites. Priority settings may be changed on a particular device to over-ride Best Master Clock selection behavior.

**Table 5B-2-23 Default PTP Clock Settings**

| PTP Clock Attribute | Controller | I/O Device | Switch or Router | Primary Source |
|---|---|---|---|---|
| Clock Type(s) | Ordinary<br>(Master / Slave) | Ordinary<br>(Slave Only) | Boundary and/or<br>Transparent | Ordinary<br>(Master Only) |
| Clock Class | 248 | 255 | 248 | 248 |
| Clock Accuracy | Unknown | unknown | Unknown | Unknown |
| Clock Variance | Device specific | Device specific | Device specific | Device specific |
| Time Source | oscillator | oscillator | oscillator | oscillator |
| Priority1 | 128 | 128 (not settable) | 128 | 128 |
| Priority2 | 128 | 128 (not settable) | 128 | 128 |

### 5B-2.15.9 TTL – Time to Live (EtherNet/IP only)

By default, the TTL value in the UDP/IPV4 packet shall be set to 1 on EtherNet/IP networks. This is required to prevent PTP packets from traversing routers which could significantly increase packet jitter. Where required, PTP time can be made to cross subnets by using routers with 1588 boundary clocks.

### 5B-2.15.10 Handset Clock Quality Management

Some CIP Sync grandmaster clocks allow the time to be set directly by the user or through a user administered mechanism. For example, a device may provide a configuration option to set the System Time from a personal computer (PC). This behavior is characterized as a HAND set clock. The process of hand setting a clock should improve the quality of the clock, since the new time will presumably be a more accurate reference. A clock may for example powerup with an unknown time. Once set the time is known and to within a specified tolerance of UTC time.

Table 5B-2.24 defines PTP settings for a clock that supports HAND set behavior. These settings are defined such that all clocks implementing HAND set behavior will interoperate. The clock class and accuracy combinations are identified along with the time source. The time source is an information only identifier and does not factor in the choice of best master.

**Table 5B-2-24 HAND Set Clock Quality Management**

| Clock Condition | Clock Class | Clock Accuracy | Time Source |
|---|---|---|---|
| Immediately after hand set. | Degradated Reference (B)<br>(187) | Within 10 seconds<br>(0x30) | HAND set<br>(0x60) |
| Powerup out of the box | Default<br>(248) | Unknown<br>(0xFE) | Oscillator<br>(0xA0) |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Note: In the table, a degraded reference B (187) is chosen for clock class instead of a primary reference (6) or Holdover reference (7) class because the Degraded reference B allows the clock to be a slave when a better master is chosen. Specifically this means the HAND set clock will synchronize its clock to the current grandmaster clock rather than going into a PASSIVE master state. Additionally, no consideration is given to aging the clock quality over time – e.g. decaying to a lower quality clock as the accuracy drifts over time, but such behavior is acceptable.

### 5B-2.15.11 Device Level Ring and Path Reconfiguration

Devices that support ring topologies and require high synchronization accuracy, such as CIP Motion, shall support automatic path re-measurement whenever there is a change in the ring topology. These devices will either directly monitor the network for a change in the network topology or indirectly be notified by other nodes that are monitoring the network. Both DLR Gateway and DLR end nodes shall implement the CIP Sync Path Reconfiguration Signaling Message (section 5B-2.15.11.1). Refer to Volume 2 , EtherNet/IP Adaptation of CIP, Section 9.5.6.3, Device Level Ring (DLR) for more information regarding DLR and CIP Sync support.

In DLR, a topology change requires that CIP Sync slave devices re-compute the path delay measurement to the master when the path changes. For example, a break in the ring will cause the path and the corresponding path measurement between the master and the slave to change. It is recommended that this computation be completed as soon as possible, within the next few sync cycles, after a network reconfiguration is detected.

A PTP slave device detects that the path has changed by one of two mechanisms. A slave device in the ring will detect that the ring has broken as indicated by the ring status of the ring protocol. A slave device outside the ring will detect the path has changed when it receives a CIP Sync Path Reconfiguration Signaling Message from the bridge node.

DLR Gateway devices that bridge between a ring and external network should generate the Signaling message on the non-ring bridged network when a ring break is detected and the master resides within the ring.

Both of these scenarios are illustrated in Figure 5B-2.2. The path between master M and slave S1 before the ring break is the path labeled 1. After the ring break the path is 2 + 3 + 4. Node S1 will detect the ring break and re-measure the path. The path between master M and slave S2 before the ring break is 1 + 2 + 3 + 5. After the ring break the path is 4 + 5. The DLR Gateway device between paths 4 and 5 will detect the ring break and signal node S2 to re-measure the path.

**Figure 5B-2.2 Path Reconfiguration in a Ring Topology**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5B-2.15.11.1 Path Reconfiguration Signaling Message

The Path Reconfiguration Signaling Message is issued by a device to notify that a network reconfiguration has been detected. The message is a CIP Sync profile specific Signaling message. DLR Gateway devices will issue this message when a network reconfiguration has been detected. DLR end nodes will receive this message and recalculate the mean path delay using the delay request-response mechanism. Signaling messages in general are defined in the IEEE 1588-2008 specification and the specific Path Reconfiguration Signaling Message is specified in Table 5B-2.25 below. Refer to previous section for application of this message.

**Table 5B-2-25 Path Reconfiguration Signaling Message**

| Signaling TLV | Field Description | Field Value |
|---|---|---|
| TLV Type | Organization Extension | 03 |
| Organization ID | ODVA CIP Sync ID | 00 21 6C |
| TLV Subtype | Path Reconfiguration | 01 |

### 5B-2.16 CIP Sync Clock Model

CIP Sync defines an offset clock model to address the requirements for industrial control applications. This model is needed because, even though the 1588 PTP protocol defines a mechanism for distributing and synchronizing time, it does not define a mechanism to compensate for step changes in time that may occur at the grandmaster clock source. These changes may occur due to one or more of the following conditions:

1. The user adjusts the master clock whose type is “HAND” set.
2. A master with a more accurate clock becomes available (new grandmaster). This may occur during system startup or after the system has been running for some time.
3. The time master is temporarily disconnected from the slave clock and then re-connected. In this situation, given any discrepancy in time between the master and the slave, a step change will occur.

Those applications requiring a stable clock in the presence of step changes in time should implement their PTP clock as a local clock, synchronized to the PTP master frequency, but not to the PTP master’s absolute time value.

In this model, the PTP protocol is used to discipline the local clock so that it ticks at the same rate as the master. The device then maintains an offset between the local clock time and system time. A small delta change in time will cause the device to make a small adjustment to the offset and to continue to “tune” its clock. A large change in time will cause the device to update its offset but not “tune” its clock.

Cyclic events may then be scheduled based on the local clock and will not be affected by large step changes in time -- provided that the local clock remains synchronized to the PTP master clock frequency.

Figure 5B-2.3 shows the relationship between the Local Clock, System Time Offset, and the System Time for a PTP Time Slave. The “system to local clock offset” is a memory variable maintained by the PTP clock implementation to make offset adjustments. A leap second value is added to the offset to give System Time in terms of the proper epoch.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5B-2.3 CIP Sync Offset Clock Model**

Figure 5B-2.4 shows a system of devices each implementing the CIP Sync Offset Clock Model. Each device is a part of a network of devices that share the same concept of System Time. Each device also has a Local Clock value based on a frequency-disciplined timer and related to System Time via an offset value (System Time Offset). Such a system allows each device to maintain a local time independent from all other devices, but share a common notion of system time. As such, System Time may change without requiring changes to the local clock.

**Figure 5B-2.4 CIP Sync System with Offset Clock Model**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-2.17 CIP Sync System Time Compensation

An application creates a timestamp by reading the clock that has been synchronized by PTP and making any required adjustments to the value, for example, converting it to System Time. Additional adjustments may be required if a step has occurred in System Time.

CIP Sync defines a mechanism to maintain a consistent set of timestamps in the presence of step changes to System Time. A step change in System Time effectively causes a shift in the time base of the system. Any step changes to the grandmaster clock time must propagate through the system. Since, all nodes in the system will not see the step change at the same instance of time, a timestamp taken on one node may be inconsistent with a timestamp on another node. Or a timestamp taken at one instance in time may be inconsistent with a timestamp taken later in time on the same node. A compensation algorithm is needed to make timestamps consistent with each other before they are used in computations.

Two possible algorithms are described in the following section.

The decision to compensate timestamps is made based on the requirements of the application.

### 5B-2.17.1 Step Compensation Algorithms

An offset value is maintained with each timestamp. This offset value is the System Time Offset at the time the timestamp is captured. The offset can be used to provide an indication of when a step occurs in System Time. If the timestamp is transmitted across the network from one node to a second node the offset is sent along with the timestamp. If the two timestamps are captured within the same node, the offsets are stored along with the timestamps.

The algorithms transform a timestamp from one time base to a second time base if a time base change occurs between the time the two timestamps are captured. When the two timestamps are compared they will reference the same time base. These algorithms may equally be applied to a set of timestamps.

#### 5B-2.17.1.1 Compensation Algorithm for Timestamps within a Single Node

This algorithm is defined to allow a node to adjust the value of one or more timestamps that have been captured on the local node.

The algorithm is stated as follows:

$Timestamp_C = Timestamp_0 + Offset_1 – Offset_0$

Where,

$Timestamp_C$ is the compensated timestamp

$Timestamp_0$ is the timestamp to be compensated

$Offset_1$ is the offset associated with the timestamp at the target time base

$Offset_0$ is the offset for the timestamp to be compensated

The example in the following table compares two timestamps T1 and T2. At time T2 a step of 1000 occurs. The timestamp at T1 is adjust to 1200 to be consistent with T2 for comparison.

**Figure 5B-2.5 Example: Compensated Timestamp on a Single Node**

| Index | Timestamp | Offset |
|---|---|---|
| T0 | 100 | 0 |
| T1 | 200 | 0 |
| T2 | 1300 | 1000 |
| T3 | 1400 | 1000 |

Step

Compensate Timestamp at time T1

$Timestamp_{(T1)} = Timestamp_{(T1)} + Offset_{(T2)} – Offset_{(T1)}$

$Timestamp_{(T1)} = 200 + 1000 – 0 = 1200$


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-2.17.1.2 Compensation Algorithm for Timestamps between Nodes

This algorithm is defined to allow a node to adjust the value of a received timestamp from a remote source node so that it can be compared to a timestamp on its own node, the destination node.

The destination node must be able to detect a step change to the System Time on the remote node or on its own node and make the appropriate adjustment. Two conditions are possible:

1. The source device has seen a step change in time but the destination device has not or
2. The destination device has seen a step change in time but the source device has not.

The step change is detected by a change in the SystemTimeOffset by either the source or destination. The source offset is sent to the destination device along with the timestamp. The destination device compares the offset received to the previously received offset to determine if a step change has occurred and adjusts the received timestamp value accordingly.

The algorithm is stated as follows:

$Timestamp_{Compensated} = Timestamp_{Received} + ((Dest_{Offset} – Dest_{LastOffset}) – (Source_{Offset} – Source_{LastOffset}))$

Where:

$Timestamp_{Received}$ is received timestamp.

$Dest_{Offset}$ is the current value of the local clock time offset at the destination.

$Dest_{LastOffset}$ is the previous value of the local clock time offset at the destination.

$Source_{Offset}$ is the received value of the local clock time offset from the source.

$Source_{LastOffset}$ is the previous value of the local clock time offset from the source.

Last offsets are updated when:

$$|(Dest_{Offset} – Dest_{LastOffset}) – (Source_{Offset} – Source_{LastOffset})| <= SyncBoundsLimit$$

Where:

SyncBoundsLimit is a relatively small number that defines that synchronization bounds for the application.

The example in the following table compares two timestamps between nodes after a system time step of 1000. For illustration purposes, a packet is assumed to be sent at the sync interval with a propagation delay of 0; the offsets are arbitrarily assumed to be the same and initially 0, and the syncBoundsLimit is assumed to be a number smaller than 1000.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5B-2.6 Example: Compensated Timestamp Between Nodes**

| Interval | Master Time | Source Node | | | | | Destination Node | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Local Clock | Offset | Last Offset | Sys Time | Sent Time stamp | Local Clock | Offset | Last Offset | Sys Time | Recv Time stamp |
| T1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 100 | 100 | 0 | 0 | 100 | 100 | 100 | 0 | 0 | 100 | 100 |
| T3 | 200 + 1000 | 200 | 0 + 1000 | 0 | 1200 | 1200 | 200 | 0 | 0 | 200 | 200 |
| T4 | 1300 | 300 | 1000 | 0 | 1300 | 1300 | 300 | 0 + 1000 | 0 | 1300 | 1300 |
| T5 | 1400 | 400 | 1000 | 1000 | 1400 | 1400 | 400 | 1000 | 1000 | 1400 | 1400 |

Step

Adjusted Time

At interval T3 a step change of 1000 occurs in the master clock and is seen by the source node clock. The timestamp received at the destination clock will be compensated for by 1000 and adjusted to a value consistent with the destination clock which has not yet seen the step change (recv timestamp adjusted from 1200 to 200). At interval T4 the destination node sees the step change but no adjustment is made to the timestamp because the source and destination compensations cancel out. After interval T4 and shown in interval T5, the last offset values are updated.

A comparison of the compensated received timestamp and the current time system time will yield a meaningful and consistent result (200 and 200 vs. 1200 and 200).

## 5B-2.18 CIP Sync Group Startup Sequence

CIP Sync defines a group startup sequence and a group synchronizing service. The group startup sequence allows a group of devices to guarantee that their clocks have all been synchronized to the PTP master reference clock before starting an application that depends on System Time. The application defines the specific requirements needed for the group to be considered synchronized.

Each application (object) that wishes to synchronize to a group will implement the GroupSync service. The GroupSync service is a CIP Common service. See Appendix A.

An application may send additional parameters as part of the group sync service. For example, it may exchange the SystemTimeOffset attribute to facilitate timestamp compensation, or a period and phase to initiate a synchronized periodic event, or one or more bounding parameters to specify a target synchronization requirement.

The service returns true if the device is synchronized to the PTP Time master otherwise it returns false.

An example startup sequence for a group of applications that need to synchronize is illustrated in the figure below.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Once the controller itself is synchronized to the master time clock, it initiates a series of GroupSync messages to each of the devices also part of the same group. Each device returns a response indicating whether it is also synchronized with the PTP master reference clock. If the controller and all devices are synchronized then the group is synchronized. Otherwise, the master application repeats the synchronizing sequence or takes some fault action.

The Sync‘d status of each node is determined by the requirements of the application and may be contingent on additional synchronization parameters.

**Figure 5B-2.7 Example Startup Sequence**

## 5B-2.19 Time Sync Status Indication

A device shall provide indication of clock synchronization status. Table 5B-2.26 shows the suggested synchronization status for various indicators.

A device is considered synchronized when its local clock is synchronized with the master reference clock. However, an application may impose more stringent synchronization requirements.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-2-26 Time Sync Status Indication**

| Visual Status Indicator | Suggested Operation |
|---|---|
| Module Status LED | Standby (flashing green) = not synchronized<br>Operational (Solid green) = synchronized + other device requirements<br>Note: See module status indicator definition in the appropriate CIP network adaptation specification. |
| Multi-character Alpha-Numeric Display | A device waiting for synchronization will display “Synching”<br>A device that is a PTP Master will display “Time Master”<br>A device that is a PTP Grandmaster will display “Time GM” |
| Web or Property Page (suggested) | Identity of the grandmaster, parent and local clock<br>Grandmaster and local clock quality<br>Synchronization status<br>Current system time<br>Current offset to master<br>Current port status |

## 5B-2.20 Instance Attribute Changes from revision 1 to revision 2

The following table shows the attribute changes from version 1 to version 2. For additional detail on version 1, refer to Edition 3.2 of this specification.

**Table 5B-2-27 Time Sync Object Instance Attribute Changes from version 1 to version 2**

| Attr ID | Version 1 Name | Data Type | Version 2 Name | Data Type | Notes |
|---|---|---|---|---|---|
| 1 | EnablePTP | BOOL | PTPEnable | BOOL | Name change |
| 2 | IsSynchronized | BOOL | IsSynchronized | BOOL | No change |
| 3 | CurrentTimeMicroseconds | LINT | SystemTimeMicroseconds | ULINT | Name change<br>Data type change |
| 4 | CurrentTimeNanoseconds | LINT | SystemTimeNanoseconds | ULINT | Name change<br>Data type change |
| 5 | OffsetToMaster | LINT | OffsetFromMaster | LINT | Name change |
| 6 | MaxOffsetToMaster | LINT | MaxOffsetFromMaster | ULINT | Name change<br>Data type change |
| 7 | DelayToMaster | LINT | MeanPathDelayToMaster | LINT | Name change |
| 8 | GrandMasterClockInfo | STRUCT of | GrandMasterClockInfo | STRUCT of | Data type change |
| | Identifier | STRING[4] | ClockIdentity | USINT[8] | |
| | Stratum | INT | ClockClass | UINT | |
| | Variance | INT | TimeAccuracy | UINT | |
| | CommunicationTechnology | INT | OffsetScaledLogVariance | UINT | |
| | PortId | INT | CurrentUtcOffset | UINT | |
| | UUID | ARRAY of SINT[6] | TimePropertyFlags | WORD | |
| | | | TimeSource | UINT | |
| | | | Priority1 | UINT | |
| | | | Priority2 | UINT | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Time Sync Object, Class Code: $43_{hex}$

| Attr ID | Version 1 Name | Data Type | Version 2 Name | Data Type | Notes |
|---|---|---|---|---|---|
| 9 | ParentClockInfo | STRUCT of | ParentClockInfo | STRUCT of | Data type change |
| | Reserved | DINT | ClockIdentity | USINT[8] | |
| | ObservedDrift | DINT | PortNumber | UINT | |
| | ObservedVariance | INT | ObservedOffsetScaledLog Variance | UINT | |
| | Variance | INT | ObservedPhaseChangeRate | UDINT | |
| | CommunicationTechnology | INT | | | |
| | PortId | INT | | | |
| | UUID | ARRAY of SINT[6] | | | |
| 10 | LocalClockInfo | STRUCT of | LocalClockInfo | STRUCT of | Data type change |
| | Identifier | STRING[4] | ClockIdentity | USINT[8] | |
| | Stratum | INT | ClockClass | UINT | |
| | Variance | INT | TimeAccuracy | UINT | |
| | CommunicationTechnology | INT | OffsetScaledLogVariance | UINT | |
| | PortId | INT | CurrentUtcOffset | UINT | |
| | UUID | ARRAY of SINT[6] | | | |
| 11 | NumberOfPorts | SINT | NumberOfPorts | UINT | Data type change |
| 12 | PortState | STRUCT of | PortStateInfo | STRUCT of | Name change<br>Data type change |
| | | INT | NumberOfPorts | UINT | |
| | | ARRAY of SINT | | ARRAY of STRUCT | |
| | | | PortNumber | UINT | |
| | | | PortState | UINT | |
| 13 | PortEnable | STRUCT of | PortEnableCfg | STRUCT of | Name change<br>Data type change |
| | | INT | NumberOfPorts | UINT | |
| | | ARRAY of SINT | | ARRAY of STRUCT | |
| | | | PortNumber | UINT | |
| | | | PortEnable | UINT | |
| 14 | PortBurstEnable | | PortLogAnnounceInterval Cfg | See attrib 14 | Replaced attribute |
| 15 | SyncInterval | SINT | PortLogSyncIntervalCfg | UINT | Name change<br>Data type change |
| | | | | ARRAY of STRUCT | |
| | | | PortNumber | UINT | |
| | | | PortLogSyncInterval | UINT | |
| 16 | PreferredMaster | | Priority1 | See attrib 16 | Replaced attribute |
| 17 | Subdomain | | Priority2 | See attrib 17 | Replaced attribute |
| 18 | ClockMode | | DomainNumber | See attrib 18 | Replaced attribute |
| 19 | StepsRemoved | | ClockType | See attrib 19 | Replaced attribute |
| 20 | SystemTimeOffset | | ManufactureIdentity | See attrib 20 | Replaced attribute |
| 21 | | | ProductDescription | See attrib 21 | New attribute |
| 22 | | | RevisionData | See attrib 22 | New attribute |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Time Sync Object, Class Code: $43_{hex}$

| Attr ID | Version 1 Name | Data Type | Version 2 Name | Data Type | Notes |
|---|---|---|---|---|---|
| 23 | | | UserDescription | See attrib 23 | New attribute |
| 24 | | | PortProfileIdentityInfo | See attrib 24 | New attribute |
| 25 | | | PortPhysicalAddressInfo | See attrib 25 | New attribute |
| 26 | | | PortProtocolAddressInfo | See attrib 26 | New attribute |
| 27 | | | StepsRemoved | See attrib 27 | New attribute |
| 28 | | | SystemTimeAndOffset | See attrib 28 | New attribute |
| 29 | | | TTLSetting | | New attribute |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5B-3 Modbus Object

## Class Code: 44hex

The Modbus object provides an interface to Modbus data existing in a CIP device.

Services are provided that directly correspond to the most common Modbus commands. For other Modbus commands a passthrough service is provided to directly pass those commands through the CIP interface to the Modbus sub-system within the device.

The Modbus Object is always associated with a CIP device. When it is used in a CIP to Modbus translator device it provides a proxy between the CIP device it is associated with and the Modbus device. When used in a CIP device that has internal Modbus characteristics, it provides a CIP interface to these characteristics.

A CIP device with the Modbus Object would use the device type for the functionality associated with the device and not the Modbus device type (0x28).

Note: This object is not the CIP to Modbus translation function which is defined in Volume 7. The use of this object in a device with the CIP to Modbus translation function is described in Volume 7 - The Integration of Modbus Devices to the CIP Architecture.

**Figure 5B-3.1 An example of a former Modbus device enhanced with CIP objects to become a CIP device**

## 5B-3.1 Class Attributes

| Number | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 | | | | | |

## 5B-3.2 Instance Attributes

| Number | Need in Implem | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| No Instance Attributes are defined | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-3.3 Common Services

The Modbus Object provides the following Common Services:

**Table 5B-3-1 Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0E hex | Conditional [1] | N/A | Get_Attribute_Single | Returns the contents of the specified attribute |

Table Footnotes:
1 This service is required if any of the class attributes are supported

## 5B-3.4 Object-specific Services

The Modbus Object provides the following Object-specific services:

**Table 5B-3-2 Configuration of the Object-specific Services**

| Service Code | Need in Implem | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 4B hex | N/A | Conditional [1, 3] | Read_Discrete_Inputs | Reads one or more contiguous discrete input(s). |
| 4C hex | N/A | Conditional [1, 3] | Read_Coils | Reads one or more contiguous coil(s). |
| 4D hex | N/A | Conditional [1, 3, 4] | Read_Input_Registers | Reads one or more contiguous input register(s). |
| 4E hex | N/A | Conditional [1, 3, 4] | Read_Holding_Registers | Reads one or more contiguous holding register(s). |
| 4F hex | N/A | Conditional [1, 3] | Write_Coils | Writes one or more contiguous coil(s). |
| 50 hex | N/A | Conditional [1, 3, 4] | Write_Holding_Registers | Writes one or more contiguous holding register(s). |
| 51 hex | N/A | Optional [2] | Modbus_Passthrough | Provides encapsulation of any public or private Modbus function. |

Table Footnotes:
1 The service shall be supported if the associated Modbus function code and/or data item is supported by the device.
2 When the Modbus Passthrough service is present it may be used for the same functions as the CIP services already provided.
3 The parameter data of these services shall be little-endian. Since Modbus data is big-endian, the Modbus Object may have to swap bytes when sending the message to the Modbus subsystem.
4 The response data of these services shall be little-endian. Since Modbus data is big-endian, the Modbus Object may have to swap bytes after receiving a response from the Modbus subsystem.

### 5B-3.4.1 Read_Discrete_Inputs Service

This service reads one or more discrete inputs from the Modbus Discrete Inputs table. This service is the equivalent of a Read Discrete Inputs function (function code 0x02) on Modbus.

**Table 5B-3-3 Read_Discrete_Inputs Request Service Parameters**

| Name | Data Type | Description of Request Parameter | Semantics of Values |
|---|---|---|---|
| Starting Address | UINT | Offset in table to begin reading from.[1] | Zero based |
| Quantity of Inputs | UINT | Number of inputs to read[1] | |

Table Footnotes:
1 The Request Parameter is little endian. The Modbus protocol is big endian. The implementer may have to swap bytes depending on the Modbus subsystem implementation.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-3-4 Read_Discrete_Inputs Response Service Parameters**

| Name | Data Type | Description of Response Parameter | Semantics of Values |
|---|---|---|---|
| Input Values | ARRAY of octet | Input values read. Each input is packed as a bit within a byte. | |

### 5B-3.4.2 Read_Coils Service

This service reads one or more coils from the Modbus Coils table. This service is the equivalent of a Read Coils function (function code 0x01) on Modbus.

**Table 5B-3-5 Read_Coils Request Service Parameters**

| Name | Data Type | Description of Request Parameter | Semantics of Values |
|---|---|---|---|
| Starting Address | UINT | Offset in table to begin reading from.[1] | Zero based |
| Quantity of Coils | UINT | Number of coils to read [1] | |

Table Footnotes:
1 The Request Parameter is little endian. The Modbus protocol is big endian. The implementer may have to swap bytes depending on the Modbus subsystem implementation.

**Table 5B-3-6 Read_Coils Response Service Parameters**

| Name | Data Type | Description of Response Parameter | Semantics of Values |
|---|---|---|---|
| Coil Status | ARRAY of octet | Coil values read. Each input is packed as a bit within a byte. | |

### 5B-3.4.3 Read_Input_Registers Service

This service reads one or more input registers from the Modbus Input Register table. This service is the equivalent of a Read Input Registers function (function code 0x04) on Modbus

**Table 5B-3-7 Read_Input_Registers Request Service Parameters**

| Name | Data Type | Description of Request Parameter | Semantics of Values |
|---|---|---|---|
| Starting Address | UINT | Offset in table to begin reading from.[1] | Zero based |
| Quantity of Input Registers | UINT | Number of input registers to read [1] | |

Table Footnotes:
1 The Request Parameter is little endian. The Modbus protocol is big endian. The implementer may have to swap bytes depending on the Modbus subsystem implementation.

**Table 5B-3-8 Read_Input_Registers Response Service Parameters**

| Name | Data Type | Description of Response Parameter | Semantics of Values |
|---|---|---|---|
| Input Register Values | ARRAY of 16-bit word[1] | Input register values read[1,2] | |

Table Footnotes:
1 The data is returned as 16 bit entities for each register. The actual data type of the values is unknown.
2 The Response Data is little endian. The Modbus protocol is big endian. The implementer may have to swap bytes depending on the Modbus subsystem implementation.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-3.4.4 Read_Holding_Registers Service

This service reads one or more holding registers from the Modbus Holding Register table. This service is the equivalent of a Read Holding Registers function (function code 0x03) on Modbus. The service parameter data shall be received in little-endian format. The response data shall be an array of 16-bit words sent as a little-endian value. Because Modbus data is big-endian, the Object may have to swap bytes when sending/receiving a Read Holding Register function message to/from the Modbus subsystem.

**Table 5B-3-9 Read_Holding_Registers Request Service Parameters**

| Name | Data Type | Description of Request Parameter | Semantics of Values |
|---|---|---|---|
| Starting Address | UINT | Offset in table to begin reading from.[1] | Zero based |
| Quantity of Holding Registers | UINT | Number of holding registers to read [1] | |

Table Footnotes:
1 The Request Parameter is little endian. The Modbus protocol is big endian. The implementer may have to swap bytes depending on the Modbus subsystem implementation.

**Table 5B-3-10 Read_Holding_Registers Response Service Parameters**

| Name | Data Type | Description of Response Parameter | Semantics of Values |
|---|---|---|---|
| Holding Register Values | ARRAY of 16-bit word [1] | Holding register values read [2] | |

Table Footnotes:
1 The data is returned as 16 bit entities for each register. The actual data type of the values is unknown.
2 The Response Data is little endian. The Modbus protocol is big endian. The implementer may have to swap bytes depending on the Modbus subsystem implementation.

### 5B-3.4.5 Write_Coils Service

This service writes one or more coils to the Modbus Coils table. This service is the equivalent of a Write Multiple Coils function (function code 0x0F) on Modbus. If the Quantity of Outputs field is 1, it can be considered a Write Single Coil function (function code 0x05).

**Table 5B-3-11 Write_Coils Request Service Parameters**

| Name | Data Type | Description of Request Parameter | Semantics of Values |
|---|---|---|---|
| Starting Address | UINT | Offset in table to begin writing to [1] | Zero based |
| Quantity of Outputs | UINT | Number of output coils to write [1] | |
| Output Values | ARRAY of octet | Output coil values | |

Table Footnotes:
1 The Request Parameter is little endian. The Modbus protocol is big endian. The implementer may have to swap bytes depending on the Modbus subsystem implementation.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-3-12 Write_Coils Response Service Parameters**

| Name | Data Type | Description of Response Parameter | Semantics of Values |
|---|---|---|---|
| Starting Address | UINT | Offset in table where writing began [1] | Zero based |
| Quantity of Outputs | UINT | Number of output coils written [1] | |

Table Footnotes:
1 The Response Parameters are little endian. The Modbus protocol is big endian. The implementer may have to swap bytes depending on the Modbus subsystem implementation.

### 5B-3.4.6 Write_Holding_Registers Service

This service writes one or more holding registers to the Modbus Holding Registers table. This service is the equivalent of a Write Multiple Registers function (function code 0x10) on Modbus. If the Quantity of Outputs field is 1, it can be considered a Write Single Register function (function code 0x06).

**Table 5B-3-13 Write_Holding_Registers Request Service Parameters**

| Name | Data Type | Description of Request Parameter | Semantics of Values |
|---|---|---|---|
| Starting Address | UINT | Offset in table to begin writing to [2] | Zero based |
| Quantity of Outputs | UINT | Number of output registers to write [2] | |
| Output Values | ARRAY of 16-bit word [1] | Output register values | |

Table Footnotes:
1 The data is returned as 16 bit entities for each register. The actual data type of the values is unknown
2 The Request Parameter is little endian. The Modbus protocol is big endian. The implementer may have to swap bytes depending on the Modbus subsystem implementation.

**Table 5B-3-14 Write_Holding_Registers Response Service Parameters**

| Name | Data Type | Description of Response Parameter | Semantics of Values |
|---|---|---|---|
| Starting Address | UINT | Offset in table where writing began[1] | Zero based |
| Quantity of Outputs | UINT | Number of output registers written [1] | |

Table Footnotes:
1 The Response Parameters are little endian. The Modbus protocol is big endian. The implementer may have to swap bytes depending on the Modbus subsystem implementation.

### 5B-3.4.7 Modbus_Passthrough Service

The Modbus_Passthrough service provides a way for a client to initiate a specific Modbus function. The Modbus request and response are encapsulated in the CIP request and response service data fields with no modification. This service shall not attempt to perform any endian conversion on the bytes in the data stream (either request or response).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

To improve or provide interoperability with the greatest number of CIP clients this service should not be used to pass through Modbus commands that are otherwise accessible by CIP object specific services 4Bhex through 50hex. The pass through service should not be implemented as a substitute for the CIP object specific services 4Bhex through 50hex.

**Table 5B-3-15 Modbus_Passthrough Request Service Parameters**

| Name | Data Type | Description of Request Parameter | Semantics of Values |
|---|---|---|---|
| function_code | USINT | Function code of the Modbus request | 1-127 |
| request_data | ARRAY of octet | Parameter data for the Modbus function request. This may include sub-function codes | Maximum of 252 bytes |

There are two possible responses to any Modbus request. The first is a normal response and its contents are shown in Table 5B-3.16. If an error is detected executing a Modbus request, an exception response is returned, as shown in Table 5B-3.17. In this case, the most significant bit of the function_code is set to indicate an exception response.

**Table 5B-3-16 Modbus_Passthrough Response Service Parameters – Normal Response**

| Name | Data Type | Description of Response Parameter | Semantics of Values |
|---|---|---|---|
| function_code | USINT | Function code of the Modbus request | 1-127 |
| response_data | ARRAY of octet | Parameter data for the Modbus function response. This may include sub-function codes | Maximum of 252 bytes |

**Table 5B-3-17 Modbus_Passthrough Response Service Parameters – Exception Response**

| Name | Data Type | Description of Response Parameter | Semantics of Values |
|---|---|---|---|
| exception-function_code | USINT | Exception function code for the Modbus request (function_code + 128) | 129-255 |
| exception_code | ARRAY of octet | Exception response code | Maximum of 252 bytes |

## 5B-3.5 Error Checking

The following conditions will result in a General Error Code of 0x20 (Invalid Parameter):

- Requesting zero (0) quantity of data values
- Requesting more data values than can be supported by Modbus; for example requesting to read 500 holding registers.
- Any request where combining the start of the data offset with the quantity of data would cause an overflow, for example requesting to read 100 registers starting at offset hex 0xFFFF.

The following condition will result in a General Error Code of either 0x13 (Not Enough Data) or 0x15 (Too Much Data):

- Write requests where the bytes of attached data do not properly match the quantity of data values being written; for example, requesting to write 4 registers but supplying any number of data bytes other than 8


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5B-4 Originator Connection List Object

**Class Code: 45 Hex**

The Originator Connection List Object may be implemented in devices that originate connections. Devices that do not originate connections shall not support the Originator Connection List object. The Originator Connection List Object can be used by clients to read the set of connection information the device is configured to originate. The Originator Connection List object is independent of the mechanism used to configure the device that originates connections.

A client can filter the set of connections to be returned in response to the Connection Read service (0x4C) to connections that traverse a specific network or device.

The Port object shall be supported if the Originator Connection List object is supported and the device supports more than one CIP Port or the device supports a single CIP Port and the Port Number is not 2. The Port object is required because a client of the Originator Connection List object needs to be able to learn the Port Numbers of the incoming port for the Create service.

## 5B-4.1 Class Attributes

The Originator Connection List Object class attributes are specified in Table 5B-4.1.

**Table 5B-4-1 Originator Connection List Object Class Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Value |
|---|---|---|---|---|---|---|---|
| 1 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | | |

## 5B-4.2 Instance Attributes

The Originator Connection List Object instance attributes are specified in Table 5B-4.2.

**Table 5B-4-2 Originator Connection List Object Instance Attributes**

| Attr ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Value |
|---|---|---|---|---|---|---|---|
| There are no instance attributes. | | | | | | | |

## 5B-4.3 Services Supported

The required services shall be as specified in Table 5B-4.3.

**Table 5B-4-3 Services at Class Level and Instance Level**

| Service | Needed in Implementation | | Service | Description of |
|---|---|---|---|---|
| Code | Class | Instance | Name | Service |
| 0x08 | Required | N/A | Create | Creates an instance of the Originator Connection List object (See the Create service definition below) |
| 0x09 | N/A | Required | Delete | Deletes an instance of the Originator Connection List object (See the Delete service definition below) |
| 0x4C | N/A | Required | Connection Read | Returns a collection of connection information (See Read service definition below) |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-4.3.1 Create (0x08)

The Create service is used to instantiate an Originator Connection List object. Within the create message, the client shall include a Path from originator specifying the route from the Connection Originator device to a device or network (a Create service parameter specifies if the Path from originator specifies a device or network). The route to the device or network shall consist of port segments. This route is used to select the set of connections to return. Only connections that traverse the specified route shall be returned.

If the Path from originator specifies a device (the link address portion of the last port segment not ignored), then all connections routed to or through the specified device will be returned. If the Path from originator specified a port (the link address of the last port segment is ignored), then all connections routed to or through the specified network will be returned.

The Originator Connection List Object shall reply with an instance number to which all further communication will be addressed for this session.

The instance shall delete itself if communications to its client are severed. Communications to an instance shall be monitored via a watchdog timer whose time-out value is 60 seconds. The watchdog timer for an instance shall be reset when it receives any message.

Table 5B-4.4 defines the request parameters for the Create service, Table 5B-4.5 defines the response parameters for the Create service, and Table 5B-4.6 defines the response status values supported by the Create service in addition to other error codes listed in Appendix B.

**Table 5B-4-4 Create Service Request**

| Parameter | Data Type | Description |
|---|---|---|
| Path from originator properties | BYTE | Bit 0 - Indicates if the Path from originator addresses a device or network. If the Path from originator addresses a network the link address of the last port segment is ignored. A value of 0 indicates the Path from originator addresses a device, a value of 1 indicates the Path from originator addresses a network.<br><br>Bits 1-7 – Reserved by CIP, shall be set to 0. |
| Path from originator Size | USINT | The number of 16-bit words in the Path from originator parameter |
| Path from originator | Padded EPATH | Path from the originator device used to select the connections to return in response to a read (0x4C) service. |

**Table 5B-4-5 Create Service Response**

| Parameter | Data Type | Description |
|---|---|---|
| Instance | UDINT | The Originator Connection List object instance that was created |

**Table 5B-4-6 Create Service Status Descriptions**

| General Status | Extended Status | Status Name | Description |
|---|---|---|---|
| 0x02 | 0x0001 | Resource unavailable | The maximum number of Originator Connection List Object instances already exist |
| | 0x0002 | Resource unavailable | Not enough memory on device that originates connections |
| 0xD0 | None | Connection list empty | There are no connections configured based on the specified parameters |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-4.3.2 Delete (0x09)

The Delete service is used to conclude a session. It deallocates all resources for a specified instance of the Originator Connection List Object. The Delete service shall not be supported at the class level.

Table 5B-4.7 defines the optional request parameter for the Delete service. If the parameter is specified the device shall ignore it.

There are no response parameters for the Delete service.

**Table 5B-4-7 Delete Service Request**

| Parameter | Data Type | Description |
|---|---|---|
| Instance | UDINT | The Originator Connection List object instance to delete |

### 5B-4.3.3 Connection Read (0x4C)

The Connection Read service is used by a client to read information about the connections that the Connection Originator is configured to open. The set of connections are limited by the Create service parameters. The connection information comes from application object(s) (such as a Connection Configuration Object) within the device that originates connections.

A Connection Read service request shall contain a UDINT parameter that specifies the connection index on which to start the read. To read the first connection, a value of 0 is specified as the connection index. The Originator Connection List Object shall reply to a Connection Read service with as much connection information as can be fit into a reply packet. The connection indices within a packet shall be ordered from lowest to highest (gaps are allowed). The extended status field within the reply indicates whether connections with higher connection indices exist. A client can continue to submit Connection Read service requests with higher starting connection indices until it reads all the connection information that traverse the specified device/network. If the specified connection index does not exist or does not match, the next higher existing connection index that matches shall be returned.

An extended status of 0 shall indicate that more connections exist. An extended status of 1 shall indicate that all connections have been read.

The reply shall consist of a UINT indicating the number of connections described in this response followed by a series of entries each describing a connection that meets the connection request criteria (the route to the device or network).

Table 5B-4.8 defines the request parameters for the create service, Table 5B-4.9 defines the response parameters for the create service, and Table 5B-4.10 defines the response status values supported by the Connection Read service in addition to other error codes listed in Appendix B.

**Table 5B-4-8 Connection Read Service Request**

| Parameter | Data Type | Description |
|---|---|---|
| Connection Index | UDINT | The index of the next connection to read |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-4-9 Connection Read Service Response**

| Parameter | Data Type | Description |
|---|---|---|
| Number of connections | UINT | The number of connections returned in response to the Connection Read service |
| Connections | Array of Struct of | Array of connections read (number of elements is Number of connections) |
| Size | UINT | Size of the remaining Parameters beginning with the Connection index – this exists to allow additional elements to be added to the end of the Connection structure in the future |
| Connection index | UDINT | The index of this connection |
| Connection timeout multiplier | USINT | The value used for the connection timeout multiplier field the Forward_Open service request[1] |
| Transport class and trigger | BYTE | The value used for the Transport Type/Trigger field defined the Forward_Open service request[1] |
| Originator to Target Netparams | UDINT | The value used for the O→T Network Connection Parameters field of the Forward_Open service request[2] |
| Target to Originator Netparams | UDINT | The value used for the T→O Network Connection Parameters field of the Forward_Open service request[2] |
| Originator to Target RPI | UDINT | The value used for the O→T RPI field of the Forward_Open service request1 |
| Target to Originator RPI | UDINT | The value used for the T→O RPI field defined of the Forward_Open service request1 |
| Path Size | USINT | The value used for the Connection_Path_size field of the Forward_Open service request1 |
| Pad | USINT | Pad |
| Connection Path | Padded EPATH | The value used for the Connection_Path field of the Forward_Open service request[1] |

Table Footnotes:

1 See Chapter 3 for the definition of the Forward_Open service parameters

2 See Chapter 3 for the definition of the Large_Forward_Open service parameters

**Table 5B-4-10 Connection Read Service Status Descriptions**

| General Status | Extended Status | Status Name | Description |
|---|---|---|---|
| 0x00 | 0x0000 | Success | More connections to read |
| | 0x0001 | Success | No more connections to read |
| 0xD1 | None | Data set integrity error | The set of selected connections has changed since this instance was created. The client should delete this instance of the Originator Connection List object and create a new instance to read the connection list. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5B-5 Target Connection List Object

## Class Code: 4D Hex

The Target Connection List object may be implemented by any device that accepts the Forward Open services of the Connection Manager object (0x06, see section 3-5) and is used to report the I/O connection targets that the device supports.

This object supports the following use cases:

- Enable a client to configure a "connectivity test" connection to a target device. This connection shall have no impact on the device's application (i.e. field outputs). The sole purpose of this connection is to enable validation of: 1) the device's attachment to the network and 2) the configuration and operation of any supporting network components in the path to the originator.
- Enable the development of "plug and play" features in originators by providing a simple way for an originator to discover the default connections (Exclusive Owner, Input Only, etc.) for a target device, including all of the Forward Open parameters.
- Provide a way for devices to report any additional I/O connection targets that they support, including those that are dynamically created (e.g. for highly configurable or programmable devices).

Predefined instances exist (see section 5B-5.3) for the default connection target for each application type that is supported (i.e. Exclusive Owner, Input Only, Listen Only, and/or Redundant Owner). Additional instances may exist when other application paths are supported for that type (e.g. various other Assemblies, Connection Points, or Symbols). Instance attributes may be provided to report any supported variations in connection parameters (e.g. multicast/unicast, cyclic/change-of-state trigger, priority 1/2, API ranges, etc).

While the Target Connection List object is independent of the mechanism that a connection originator uses to manage its scan list, the majority of the members of the connection structure returned by the Connection Read service are the same as those defined by the Connection Configuration object's (0xF3) Set_Attributes_All service (see 5B-11.4.2).

There are no provisions for supporting direct connections to proxied modular devices (see section 7 3.7) at this time.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-5.1 Class Attributes

The Target Connection List Object class attributes are specified in Table 5B-5.1.

**Table 5B-5-1Target Connection List Object Class Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Value |
|---|---|---|---|---|---|---|---|
| 1 | Conditional[1] | Get | NV | Revision | UINT | Revision of this object | Current revision of this object = 1. |
| 2 | Required | Get | V | Max Instance | UINT | Maximum instance number of an object currently created in this class level of the device. | The largest instance number of a created object at this class hierarchy level. |
| 3 | Required | Get | V | Num Instances | UINT | Number of object instances currently created at this class level of the device. | The number of object instances at this class hierarchy level. |
| 4 thru 7 | These class attributes are optional and are described in Chapter 4 of this specification. | | | | | | |
| 8 | Required | Get | NV | Connection Data Set Revision | UINT | See section 5B-5.1.1. | Default value shall be 0. |
| 9 | Required | Get | V | Capabilities | DWORD | Indicates which connection targets the device exposes via this object and the level of detail provided for each of those targets. | See section 5B-5.1.2. |

Table Footnotes:

1 If the value is 1, then this attribute is OPTIONAL in the implementation. If the value is greater than 1, then this attribute is REQUIRED.

### 5B-5.1.1 Connection Data Set Revision (Attribute 8)

Some devices may support dynamic modification of their target connection data, including the creation and/or deletion of target connections reflected by instances of this object. This attribute provides a means for clients to determine whether any changes were made to any of the data reported by this class between successive service requests. The default value for this attribute is 0. The device shall increment this attribute by one, rolling over to 0 after 65,535 is reached, for each modification made to its Connection Data Set. The Connection Data Set is the entire set of data associated with all instances of this object.

At a minimum, clients shall verify that the Connection Data Set Revision value included in the first and last Get_Attributes_All or Connection_Read service responses of a sequence of requests are the same to ensure the integrity of its copy of the target device's Connection Data Set.

### 5B-5.1.2 Capabilities (Attribute 9)

The Capabilities attribute indicates which connection targets the device exposes via this object and the level of detail provided for each of those targets, as specified in the following table:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-5-2 Capabilities Attribute Bit Definitions**

| Bit(s) | Called | Description |
|---|---|---|
| 0 | Default Targets | Set to 1 if Default instance(s) (see section 5B-5.3) exist, 0 otherwise. |
| 1 | Additional Targets | Set to 1 if Additional instances (see section 5B-5.3) exist, 0 otherwise. |
| 2 | Unreported Targets | Set to 1 if the device has non-default connection targets but no Additional Instances of this object are provided to report them, 0 otherwise. |
| 3 | Instance Attributes | Set to 1 if the instance attributes are supported. |
| 4-31 | Reserved | Sender shall set to 0, receiver shall ignore. |

## 5B-5.2 Instance Attributes

The Target Connection List Object instance attributes are specified in Table 5B-5.3.

**Table 5B-5-3 Target Connection List Object Instance Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Value |
|---|---|---|---|---|---|---|---|
| 1 | Conditional[1] | Get | NV | O→T RPI Range | Struct of:<br>UDINT<br>UDINT | Minimum and Maximum acceptable RPI | microseconds |
| 2 | Conditional[1] | Get | NV | T→O RPI Range | Struct of:<br>UDINT<br>UDINT | Minimum and Maximum acceptable RPI | microseconds |
| 3 | Conditional[1] | Get | NV | Trigger, Transport and Application Type | DWORD | The supported transport classes, production triggers and application type | Format same as Trigger and Transport Field in EDS keyword ConnectionN in 7-3.6.10.1.1 |
| 4 | Conditional[1] | Get | NV | Connection Parameters | DWORD | Fixed/variable size, RT Header, Connection Type and Priority | Format same as Connection Parameters Field in EDS keyword ConnectionN in 7-3.6.10.1.2 |
| 5 | Conditional[1] | Get | NV | O→T Size | Struct of:<br>UINT<br>UINT | Min O→T Size<br>Max O→T Size | Size in bytes |
| 6 | Conditional[1] | Get | NV | T→O Size | Struct of:<br>UINT<br>UINT | Min T→O Size<br>Max T→O Size | Size in bytes |
| 7 | Conditional[1] | Get | NV | PIT Range | Struct of:<br>UDINT<br>UDINT | Min/Max Production Inhibit Range | Time in microseconds. Values of 0 means PIT Segment not supported |
| 8 | Conditional[1] | Get | NV | Connection Name String | SHORT_STRING | The name assigned to the connection target. | The Connection Name String shall be unique across all supported connections. See 7-3.6.10.1.10 for more info. |

Table Footnotes:

1 If any of the instance attributes are supported, then all of the instance attributes shall be supported


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-5.3 Instance Assignments

The instance assignments for the Target Connection List Object are specified in Table 5B-5.4.

**Table 5B-5-4 Instance Assignments for Target Connection List Object**

| Instance | Need in Implementation | Name | Description of Instance |
|---|---|---|---|
| 1 | Required | Connectivity Test | Data Loopback connection – see section 5B-5.3.1 |
| 2 | Conditional[1] | Exclusive Owner | Default Exclusive Owner connection |
| 3 | Conditional[1] | Input Only | Default Input Only connection |
| 4 | Conditional[1] | Listen Only | Default Listen Only connection |
| 5 | Conditional[1] | Redundant Owner | Default Redundant Owner connection |
| 6 thru 99 | Reserved | | |
| 100 thru 65,535 | Optional | Additional Connections | Non-default connections. Gaps are allowed between instances. |

Table Footnotes:

1 Required if the device has a default connection of the named application type

### 5B-5.3.1 Connectivity Test Connection - Instance #1

This instance provides a way for an originator to test its CIP connectivity to a target node by establishing an Implicit (I/O) Connection with test data in the O→T direction that shall NOT be applied by the node to its field outputs regardless of the Run/Idle indication provided in the Run/Idle header, but the target application shall loop back the O→T values in the T→O direction. In some respects it is similar to the NOP service that can be used with Explicit Messaging.

If the instance attributes are supported, this connection shall be capable of supporting all of the possible parameter variations that are present across all of the connections present in the device.

The following parameters of this instance shall be:

- Exclusive Owner
- Name = Connectivity Test
- Transport Class = 1
- Fixed/Variable = Fixed
- Redundant Owner = 0
- Real Time Format O→T = Run/Idle header
- Real Time Format T→O = Modeless
- O→T Size = 8 bytes, see Table 5B-5.5 for the structure
- T→O Size = 12 bytes, see table Table 5B-5.6 for the structure
- Application Paths = 20 4D 24 00 2C 01 20 4D 24 00 2C 02
  - No Config Path
  - Consumption path = 20 4D 24 00 2C 01
  - Production path = 20 4D 24 00 2C 02

The remaining parameters shall match the parameters of at least one other connection in the device.

## 5B-5.4 Connection Points

The class instance of the Target Connection List object supports the connection ponts defined in Table 5B-5.5 and Table 5B-5.6 for the connection described by the Connectivity Test instance (1).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-5-5 Connectivity Test Connection Point 1 (O→T) Data Structure**

| Data Type | Name | Description |
|---|---|---|
| UINT | Data0 | Originator choice – data shall NOT be applied to field outputs |
| UINT | Data1 | Originator choice – data shall NOT be applied to field outputs |
| UINT | Data2 | Originator choice – data shall NOT be applied to field outputs |
| UINT | Data3 | Originator choice – data shall NOT be applied to field outputs |

**Table 5B-5-6 Connectivity Test Connection Point 2 (T→O) Data Structure**

| Data Type | Name | Description |
|---|---|---|
| UINT | Data0 Echo | Same value as received in the Data0 element in the O→T direction[1] |
| UINT | Data1 Echo | Same value as received in the Data1 element in the O→T direction[1] |
| UINT | Data2 Echo | Same value as received in the Data2 element in the O→T direction[1] |
| UINT | Data3 Echo | Same value as received in the Data3 element in the O→T direction[1] |
| UINT | Timer | Free running timer in milliseconds, 0 in the first production with modulus 65536. |
| UINT | Counter | Counter which increments by 1 with each production (API), with the first production equal to 1 with modulus 65536. |

Table Footnotes:

1 As with all Class 1 connections, when the connection is in Idle mode, the output data shall not be applied; hence this value shall NOT be updated with the value received from the originator, rather the most recent Run value received shall be returned. If the connection has never been in Run mode, 0's shall be returned.

## 5B-5.5 Common Services

The Target Connection List object provides the following Common Services (see Appendix A):

**Table 5B-5-7 Target Connection List Object Common Services**

| Service Code | Needed in Implementation | | Service Name |
|---|---|---|---|
| | Class | Instance | |
| 0x01 | Optional | Conditional[1] | Get_Attributes_All |
| 0x0E | Required | Conditional[1] | Get_Attribute_Single |

Table Footnotes:

1 Required if instance attributes are supported, else not allowed.

### 5B-5.5.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response shall be as follows:

**Table 5B-5-8 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 |
| 2 | UINT | Max Instance | |
| 3 | UINT | Number of Instances | |
| 8 | UINT | Connection Data Set Revision | |
| 9 | DWORD | Capabilities | |

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response shall be as follows:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-5-9 Get_Attributes_All Response Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| – | UINT | Connection Data Set Revision[1] | |
| 1 | STRUCT of: | O→T RPI Range | |
| | UDINT | Minimum | |
| | UDINT | Maximum | |
| 2 | STRUCT of: | T→O RPI Range | |
| | UDINT | Minimum | |
| | UDINT | Maximum | |
| 3 | DWORD | Trigger, Transport and Application Type | |
| 4 | DWORD | Connection Parameters | |
| 5 | STRUCT of: | O→T Size | |
| | UINT | Minimum | |
| | UINT | Maximum | |
| 6 | STRUCT of: | T→O Size | |
| | UINT | Minimum | |
| | UINT | Maximum | |
| 7 | STRUCT of: | PIT Range | |
| | UDINT | Minimum | |
| | UDINT | Maximum | |
| 8 | SHORT_STRING | Connection Name String | |

Table Footnotes:

1 This number holds the contents of Class Attribute #8 and is used to detect changes to the Connection Data Set of the device. See section 5B-5.1.1.

## 5B-5.6 Object Specific Services Supported

The object specific services shall be as specified in Table 5B-5.10.

**Table 5B-5-10 Services at Class Level and Instance Level**

| Service Code | Needed in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0x4C | Required | Not Allowed | Connection Read | Returns the connection parameters for the I/O connection targets supported by the device. |

### 5B-5.6.1 Connection Read (0x4C)

The Connection Read service is used by a client to read connection structures for the set of Target Connection List object instances that the device supports. The connection structure returned is based on the member definitions of the Connection Configuration (0xF3) object's Set_Attributes_All service request.

An array of structures may be returned, containing the connection structures for as many instances as will fit in the response.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Table 5B-5.11 defines the request parameters for the Connection Read service, Table 5B-5.12 defines the response parameters for the Connection Read service, and Table 5B-5.14 defines the response status values supported by the Connection Read service. In addition, any of the other error codes listed in Appendix B may be used as necessary.

### 5B-5.6.1.1 Connection Read Request

**Table 5B-5-11 Connection Read Service Request Parameters**

| Parameter | Data Type | Description |
|---|---|---|
| Connection Specifier | UINT | This value is the ID of the first instance of this object to be read. |
| Connection Structure Exclusions | WORD | Bit 0 indicates whether the Connection Name String should be omitted from the Connection Read service response.<br>Bits 1-15 are reserved. Sender shall set to 0, receiver shall ignore. |

The Connection Read service's Connection Specifier parameter specifies the first Target Connection List instance to be read.

The Connection Structure Exclusions parameter allows the client to request that the potentially large Connection Name String member be omitted from the Connection Structure(s) returned in the response. If the bit is set to 1, then the member is omitted.

### 5B-5.6.1.2 Connection Read Response

The reply shall consist of one UINT to indicate the revision of the device's Connection Data Set, a second UINT to indicate the number of connection structures provided in the response, and then a series of connection structures using the same members as the Set_All_Attributes request of the Connection Configuration object (see Section 5B-11.4.2), with the exception of the Connection Name String (see Section 7-3.6.10.1.10), as shown in Table 5B-5.12.

The Target Connection List Object shall include as many whole connection structures as can be fit into a response packet. If any one of the device's connection structures is larger than 494[1] bytes, then the device shall support the establishment of an explicit messaging connection with a response buffer greater than or equal to the length of its largest possible Connection Read Response.

The connection structures within a packet shall be ordered from lowest to highest instance (gaps are allowed). The Connection Specifier is equivalent to the Target Connection List instance. If the requested connection does not exist, then the first connection structure returned shall correspond to the next higher existing instance or, if none, the object shall return General Status 0x16 (Object does not exist).

When connection structure(s) are returned, the object shall use General Status code 0x00 (success), plus one Additional Status word. This Additional Status shall be 0x0001 to indicate that additional instances exist or 0x0000 to indicate that no numerically higher instances exist.

A client can continue to submit Connection Read requests with higher starting Connection Specifiers until it reads connection structures for all of the Target Connection List instances that the device supports, using the Additional Status field within the reply to determine whether additional instances exist.

[1] The largest UCMM request supported on ControlNet is 498, which is the smallest, maximum UCMM size on any CIP network. The Connection Read response contains two UINTs of "header" information. The Connection Name String may be excluded from the determination of the largest whole connection structure.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-5-12 Connection Read Service Response Parameters**

| Parameter Name | | Data Type | Description | | |
|---|---|---|---|---|---|
| Connection Data Set Revision | | UINT | This number is used to detect changes to the Connection Data Set of the device. See section 5B-5.1.1. | | |
| Number of Connection Structures | | UINT | The number of Connection Structures included in this response. | | |
| Connection Structures | | Array of:<br>Struct of : | | | |
| **Connection Structure** | Connection Specifier | UINT | The instance of the object being reflected in this connection structure | | |
| | Connection Application Type, Status and Structure Format | WORD | See Table 5B-5.13 | | |
| | | | **Member Name** | **Connection Structure Format, Bit 4 (See Table 5B-5.13)** | |
| | | | | **0** | **1** |
| | Connection Flags[1] | WORD | See Section 5B-11.2.2.2 | ☑ | ☑ |
| | Target Device Id | Struct of: | Target device identification.<br>See section 5B-11.2.2.3. | ☑ | ☑ |
| | | UINT | Vendor Id | | |
| | | UINT | Device Type | | |
| | | UINT | Product Code | | |
| | | USINT | Major Revision | | |
| | | USINT | Minor Revision | | |
| | Net Connection Parameters[2] | Struct of: | See 5B-11.2.2.5. | ☑ | |
| | | USINT | conn_timeout | | |
| | | BYTE | xport_class_and_trig | | |
| | | UDINT | rpi_OT | | |
| | | UINT | net_OT | | |
| | | UDINT | rpi_TO | | |
| | | UINT | net_TO | | |
| | Connection Path[3] | Struct of: | See 5B-11.2.2.6. | ☑ | ☑ |
| | | USINT | open_path_size | | |
| | | USINT | Reserved | | |
| | | Array of UINT | open_connection_path | | |
| | Target Config Data | Struct of: | See 5B-11.2.2.10. | ☑ | ☑ |
| | | UINT | config_data_size | | |
| | | Array of USINT | config_data[4] | | |
| | Large Net Connection Parameters[2] | Struct of: | See 5B-11.2.2.13. | | ☑ |
| | | USINT | conn_timeout | | |
| | | BYTE | xport_class_and_trig | | |
| | | UDINT | rpi_OT | | |
| | | UDINT | net_OT | | |
| | | UDINT | rpi_TO | | |
| | | UDINT | net_TO | | |
| | Connection Name String | SHORT_STRING | See section 7-3.6.10.1.10. | See note 5. | See note 5. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Parameter Name | Data Type | Description |
|---|---|---|

1 The Connection bit shall be set to 0.

2 If any of this connection's parameters are dependent on an already-established connection, then compatible value(s) shall be returned, otherwise the default values for this connection shall be returned.

3 The Connection Path only includes the segments that apply to the target (e.g. the Application Path(s) required to access this connection's end points with the target); routing segments and network segments are not included. A default compatible electronic key (see C-1.4.2) with all fields set to the values contained in the Target Device Id shall be included in the Connection Path so that "plug and play" clients can provide proper keying with no additional effort.

4 The current values of the configuration data associated with this connection shall be returned.

5 Depends on Bit 0 of the Connection Structure Exclusions request parameter. See section 5B-5.6.1.1.

**Table 5B-5-13 Connection Application Type, Status and Structure Format WORD Bit Definitions**

| Bit(s) | Called | Description |
|---|---|---|
| 0-2 | Application Type | 0 – Exclusive Owner<br>1 – Input Only<br>2 – Listen Only<br>3 – Redundant Owner<br>4-7 - Reserved |
| 3 | Connection Established | Set to 1 if at least one connection is established to this target, 0 otherwise. |
| 4-7 | Connection Structure Format | Bit 4 shall be set to 1 if the connection data size requires that the Large Net Connection Params member be used instead of the Net Connection Parameters member.<br>Bits 5-7 are reserved. Sender shall set to 0, receiver shall ignore. |
| 8-31 | Reserved | Sender shall set to 0, receiver shall ignore. |

**Table 5B-5-14 Connection Read Service Additional Status Definition**

| General Status | Additional Status Word | Status Name | Description |
|---|---|---|---|
| 0x00 | 0x0000 | Success | No more connections to read |
| | 0x0001 | Success | More connections to read |
| 0x11 | Size of Largest Connection Read Response | Reply data too large | Clients must send the Connection Read request via a Class 3 connection with a T→O size equal to the size specified in the Additional Status word or use the Connection Structure Exclusions request parameter to omit members that it may not need in order for the Connection Read request to succeed. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5B-6 Base Energy Object

## Class Code: 4E Hex

The Base Energy Object acts as an "Energy Supervisor" for CIP Energy implementations. It is responsible for providing a time base for energy values, provides energy mode services, and can provide aggregation services for aggregating energy values up through the various levels of an industrial facility. It also provides a standard format for reporting energy metering results. The object is energy type independent and allows energy type specific data and functionality to be integrated into an energy system in a standard way. Multiple instances of the Base Energy Object may exist in a device. For instance, an electric power monitor may count metering pulse output transitions of a separate metering device. The count of such transitions, represented by a Base Energy Object instance, would reflect the energy consumption measured by the separate device. As another example, a device may act as a proxy for the energy consumed by a number of simple devices, where each device is associated with a separate Base Energy Object instance.

An instance of the Base Energy Object may exist as a stand-alone instance; or it may exist in conjunction with an Electrical and/or Non-Electrical Energy Object instance. If an instance of the Electrical and/or Non-Electrical Energy object is implemented in a device, it must be associated with a Base Energy Object instance.

A Base Energy Object instance can aggregate energy (in kWh) from other Base Energy Object instances or, optionally, aggregate non-electrical energy in its native units from Non-Electrical Energy Object instances. In non-electrical aggregation, the Base Energy Object instance performs the aggregation in the native non-electrical units, but stores the values in an associated Non-Electrical Energy Object instance.

## 5B-6.1 Revision History

Since the initial release of this object class definition, changes have been made that require a revision update of this object class. The table below represents the revision history.

**Table 5B-6-1 Base Energy Object Revision History**

| Revision | Reason for object definition change |
|---|---|
| 01 | Initial definition at first release of specification |
| 02 | • Redefined path size in instance attributes 12 and 14 to be in words instead of bytes.<br>• Changed data types of energy odometer attributes 7, 8 and 9 to use the newly defined signed and unsigned odometer data types. No impact on behavior, cosmetic change only.<br>• Changed the Need In Implementation property of energy odometer attributes |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-6.2 Class Attributes

**Table 5B-6-2 Base Energy Object Class Attributes**

| Attrib ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Required | Get | NV | Revision | UINT | Revision of this object | The current value assigned to this attribute is two (2) |
| 2 | Conditional[1] | Get | NV | Max Instance | UINT | Maximum instance number of an object currently created in this class level of the device | The largest instance number of a created object at this class hierarchy level. |
| 3 | Conditional[1] | Get | NV | Number of Instances | UINT | Number of object instances currently created at this class level of the device. | The number of object instances at this class hierarchy level. |
| 4 thru 7 | These class attributes are described in Volume 1, Chapter 4 of the CIP Common specification. Attributes 4 through 7 are optional. | | | | | | |

Table Footnotes

1 Class Attributes 2 and 3 are required if the Create and Delete services are supported, or if more than one instance of the Base Energy Object is supported

## 5B-6.3 Instance Attributes

**Table 5B-6-3 Base Energy Object Instance Attributes**

| Attrib ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Required | Get | NV | Energy/Resource Type | UINT | Specifies the type of energy managed by this energy instance. | 0 = Generic<br>1 = Electrical<br>2 = Non-Electrical<br>3 - 99 = Reserved<br>100 - 199 = Vendor Specific<br>200 – 65535 = Reserved<br>See section 5B-6.3.1.1 |
| 2 | Required | Get | NV | Base Energy Object Capabilities | UINT | The Energy Capabilities of the Instance. | See section 5B-6.3.1.2 |
| 3 | Required | Get | NV | Energy Accuracy | UINT | Specifies the accuracy of power and energy metering results. | Typical accuracy in 0.01 percent of reading (default) or 0.01 of other units as specified in the Energy Accuracy Basis attribute<br>0 = unknown<br>See section 5B-6.3.1.3 |
| 4 | Optional | Get/Set[1] | NV | Energy Accuracy Basis | UINT | Basis of the Energy Accuracy Attribute | 0 = Percent of Reading<br>1 = Percent of Full Scale Reading<br>2 = Absolute Error (kW)<br>See section 5B-6.3.1.4 |
| 5 | Conditional[2] | Set | NV | Full Scale Reading | REAL | Full Scale Energy Transfer Rate | Power in kW |
| 6 | Conditional[3] | Get | V | Data Status | UINT | Status of the Instance or Aggregation Data | 0 = No Errors<br>1 – 65535, See section 5B-6.3.1.6 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Base Energy Object, Class Code: $4E_{Hex}$

| Attrib ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 7 | Conditional [4] | Get | NV | Consumed Energy Odometer | ODOMETER | The consumed energy value. | Energy in kWh<br>See section 5B-6.3.1.7 |
| 8 | Conditional [5] | Get | NV | Generated Energy Odometer | ODOMETER | The generated energy value. | Energy in kWh<br>See section 5B-6.3.1.8 |
| 9 | Optional | Get | NV | Net Energy Odometer | SIGNED_ ODOMETER | The total net energy value. | Energy in kWh<br>See section 5B-6.3.1.9 |
| 10 | Conditional [6] | Get | V | Energy Transfer Rate | REAL | The time rate of energy consumption or production. | Power in kW<br>See section 5B-6.3.1.10 |
| 11 | Optional | Set | NV | Energy Transfer Rate User Setting | REAL | User setting for fixed, derived or proxy power value | Power in kW<br>See section 5B-6.3.1.11 |
| 12 | Required | Get | NV | Energy Type Specific Object Path | STRUCT of: | Path to Energy Type Specific Object instance. | See section 5B-6.3.1.12 |
| | | | | Path Size | UINT | Size of Path (in words) | |
| | | | | Path | Padded EPATH | See Appendix C for format of this field. | |
| 13 | Conditional[7] | Get | NV | Energy Aggregation Path Array Size | UINT | Number Members in the "Energy Aggregation Paths" Array | See section 5B-6.3.1.13 |
| 14 | Conditional[7] | Set<br>Member Services | NV | Energy Aggregation Paths | ARRAY of STRUCT of: | Array of Paths to Energy objects to be aggregated | See section 5B-6.3.1.13 |
| | | | | Path Size | UINT | Size of Path (in words) | |
| | | | | Path | Padded EPATH | See Appendix C for format of this field. | |
| 15 | Optional | Set | NV | Energy Identifier | STRINGI | Human readable organizational unique identification string. | The default value of this string can be the null string or a string chosen by the vendor that is instance specific |
| 16 | Optional | Set | NV | Odometer Reset Enable | BOOL | Enable the resetting of Energy Odometers via a reset service to the base energy object | Default = 0 disabled<br>See section 0 |
| 17 | Conditional [8] | Get | NV | Metering State | BOOL | Returns the metering state of the instance | 0 = Not Metering 1 = Metering |
| 18 | Conditional[9] | Get | V | Extended Data Status | UINT | Returns extended data status of the instance | See section 5B-6.3.1.16 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Base Energy Object, Class Code: $4E_{Hex}$**

| Attrib ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|

Table Footnotes

1 The access rule for Attribute 4 may be implemented as Get or Set.
2 Attribute 5, Full Scale Energy Transfer Rate, is required when Attribute 4, Energy Accuracy Basis, is set to a value of 1, “Percent of Full Scale Reading”. It is also required if the Attribute 4 access rule is implemented as a Set attribute.
3 Attribute 6 is required if Attribute 2 is set to a value of 5, “Non-Electrical Aggregation.”
4 Attribute 7 is required when Attribute 2 is a value other than 4, “Energy Value Fixed” and the instance reports consumed energy.
5 Attribute 8 is required when Attribute 2 is a value other than 4, “Energy Value Fixed” and the instance reports generated energy.
6 Attribute 10 is required only when Attribute 2 is set to a value of 4, “Energy Value Fixed”.
7 Attribute 13, Energy Aggregation Path Array Size and Attribute 14, Energy Aggregation Paths, are required if Attribute 2 is set to a value of 3, “Energy Aggregated” or 5, “Non-Electrical Aggregated”.
8 Attribute 17, Metering State, is required if Start Metering and Stop Metering Services are supported.
9 Attribute 18, Extended Data Status, is required if Attribute 2 is set to a value of 5, “Non-Electrical Aggregated”.

## 5B-6.3.1 Semantics

### 5B-6.3.1.1 Energy/Resource Type, Attribute #1

The Energy/Resource Type attribute indicates the type of energy managed by this Base Energy Object instance. Table 5B-6.4 specifies the allowable values for this attribute.

**Table 5B-6-4 Attribute #1 Values for Energy/Resource Type**

| Value | Name | Definition |
|---|---|---|
| 0 | Generic | Indicates that there is no associated energy object or that it is aggregating energy of mixed energy types |
| 1 | Electrical | Indicates that there is an associated Electrical Energy Object or that it is aggregating energy of electrical energy types |
| 2 | Non-Electrical | Indicates that there is an associated Non-Electrical Energy Object or that it is aggregating energy of non-electrical energy types |
| 3 – 99 | Reserved | Reserved by ODVA and shall not be used |
| 100 – 199 | Vendor Specific | |
| 200 – 65535 | Reserved | Reserved by ODVA and shall not be used |

The value of this attribute shall be determined in accordance with Table 5B-6.5.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-6-5 Energy/Resource Type Dependencies**

| Energy/ Resource Type | Base Energy Object Capabilities (Attr. #2) | Associated Energy Object Path | Aggregation Path |
|---|---|---|---|
| 0 (Generic) | 0 (Energy Measured)<br>1 (Energy Derived)<br>2 (Energy Proxy)<br>4 (Energy Rate Fixed) | None | None |
| | 3 (Energy Aggregated) | None | Required Collection of Base instances with Generic or mixed Resource Type |
| | 5 (Non-Electrical Aggregated) – Not allowed | NA | NA |
| 1 (Electrical) | 0 (Energy Measured)<br>1 (Energy Derived)<br>2 (Energy Proxy)<br>4 (Energy Rate Fixed) | Required – Path to Electrical Energy Object instance | None |
| | 3 (Energy Aggregated) | None | Required – Collection of Base instances with Resource Types = 1 (Electrical) |
| | 5 (Non-Electrical Aggregated) – Not allowed | NA | NA |
| 2 (Non-Electrical) | 0 (Energy Measured)<br>1 (Energy Derived)<br>2 (Energy Proxy)<br>4 (Energy Rate Fixed) | Required – Path to Non-Electrical Energy Object instance | None |
| | 3 (Energy Aggregated) | None | Required – Collection of Base instances with Resource Types = 2 (Non-Electrical) |
| | 5 (Non-Electrical Aggregated) | Required – Non-Electrical instance where native unit aggregation results are reported | Required – Uniform collection of Non-Electrical instances |

### 5B-6.3.1.2 Base Energy Object Capabilities, Attribute #2

The value indicates how “Energy Odometer” values are generated and/or how “Energy Transfer Rates” are generated or specified. Note that values 6 – 65535 are reserved by ODVA for signifying future Base Energy Object capabilities for a device.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-6-6 Attribute #2 Values for Energy Value Generation**

| Value | Name | Definition |
|---|---|---|
| 0 | Energy Measured | Indicates that the instance reports energy values that are directly measured by the device |
| 1 | Energy Derived | Indicates that the instance reports energy values that are derived indirectly by the device. Energy values may be derived from fixed nominal power values, a combination of measured and nominal values, or other combinations. |
| 2 | Energy Proxy | Indicates that the instance reports energy values as a proxy for (a) separate device(s) that may not support an instance of the Base Energy Object.. |
| 3 | Energy Aggregated | Indicates that attributes 7-9 are aggregated values, aggregated from other local or remote Energy Object Instances. If Energy Capabilities is aggregated (=3), then the Energy Aggregation Paths attribute shall be required. |
| 4 | Energy Rate Fixed | Indicates that the instance reports one or more nominal or user-defined power values that remain steady when the device is in an operating state and change infrequently relative to the rate at which the data is read from the instance. |
| 5 | Non-Electrical Aggregated | Indicates that the Base Energy Object is aggregating non-electrical energy in its native units and storing it in the associated Non-Electrical Energy Object instance. For this value, both the Associated Energy Object Path and the Energy Aggregation Paths attribute shall be required. |
| 6 - 65535 | Reserved | Reserved (shall be ignored) |

An example of Energy Measured (Base Energy Object Capabilities attribute value is 0) is a power monitor, which measures voltage, current, phase angle, etc., and calculates power and energy.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

An example of Energy Derived (Base Energy Object Capabilities attribute value is 1) is an overload relay that measured motor current, assumes a value of motor voltage, and derives a value of power and energy.

Examples of Energy Proxy (Base Energy Object Capabilities attribute value is 2):

- A controller with discrete outputs controlling external devices, each output associated with an Energy Transfer Rate User Setting (Attribute 11). The controller calculates energy based on the state of the external device and the user setting.
- A Modbus translator device that calculates energy usage based on data reported by simple devices

It is suggested that the Energy Identifier, Attribute #15, be used to identify an external device represented by a Base Energy Object instance with Energy Capabilities attribute value of Energy Proxy.

An example of Energy Rate Fixed (Energy Capabilities attribute value is 4) is an instance in a simple device that reports one or more nominal or user-defined power values that remain steady when in an operating state and change infrequently relative to the rate at which the data is read from the instance.

If the Base Energy Object Capabilities attribute value is 3, “Energy Aggregated” or 5, “Non-Electrical Aggregated”, the instance shall be capable of the following:

- Aggregating energy odometer values of the child devices correctly, including cases in which one or more energy odometer values roll over during a time interval.
- Deriving energy values from child devices which report fixed or bi-state Energy Transfer Rate (power) values
- Deriving an aggregation Energy Accuracy attribute value as a function of the contribution of each child device to the possible error in the aggregated value. Refer to the Energy Accuracy attribute semantic description.

When child devices are aggregated by more than one aggregator, users should be aware that child device data may be present in more than one aggregator. Care should be taken to avoid errors caused by this.

#### 5B-6.3.1.3 Energy Accuracy, Attribute #3

This attribute specifies the accuracy of metering energy results. By default, it specifies the accuracy of energy metering results as 100 times the percentage of reading, unless otherwise specified in the optional Energy Accuracy Basis attribute. For example, as a default, a value of 50 would indicate a device accuracy of 0.5% of reading.

The value 0 indicates that accuracy is undefined. Devices with undefined accuracy should not be specified as child devices for an aggregator.

Energy Accuracy values may change, especially in the case of aggregated energy values.

All factors that affect accuracy of energy values are to be expressed in the Energy Accuracy value. Examples of such factors are:

- Accuracy and resolution of analog to digital converters
- Sampling rates and calculation update intervals
- Network response times

The Energy Accuracy of an aggregated energy value may be derived using the following method.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

$$Aggregator\ Energy\ Accuracy = \frac{\sum_{x=1}^{n} |Error_x|}{|\sum_{x=1}^{n}(DeltaEnergy_x)|}$$

Where: *Aggregator Energy Accuracy* = Energy Accuracy attribute value of the aggregated value expressed as 100 * the percent of reading

$Error_x$ = Maximum error introduced by Child Device x

$DeltaEnergy_x$ = Interval Energy Consumption of Child Device x *or*

$DeltaEnergy_x$ = Energy Transfer Rate of Child Device x * Interval Length (hours)

The Error introduced by Child Device x may be computed according to the following table.

| Energy Accuracy Basis | $Error_x$ |
|---|---|
| 0 = percent of reading | (Energy Accuracy value / 100) * $DeltaEnergy_x$ |
| 1 = percent of full scale | (Energy Accuracy value / 100) * Full Scale Reading * Interval Length (hours) |
| 2 = absolute error | (Energy Accuracy value / 100) * Interval Length (hours) |

As an example, consider the aggregated value and accuracy of the collection of devices in the table below. The interval length is 15 minutes, or 0.25 hour.

| Device | Energy Capability | Energy Accuracy | Energy Accuracy Basis | Full Scale Reading | Delta Energy | Error |
|---|---|---|---|---|---|---|
| Plant lighting (500 kW) | Measured | 100 (1%) | 0 (% of reading) | | 125 kWh | 1.25 kWh |
| Generator (80kW) | Measured | 50 (0.5%) | 1 (% of full scale) | 100 kW | -20 kWh | 0.125 kWh |
| Motor starter (75 kW) | Derived | 500 (5%) | 0 (% of reading) | | 18.8 kWh | 0.94 kWh |
| Power supply (30kW) | Fixed | 100 (1 kW) | 2 (absolute error) | | 7.5 kWh | 0.25 kWh |

The aggregated interval Total Energy value for this collection is 125 + 20 + 18.8 + 7.5 = 171.3 kWh.

The aggregated Energy Accuracy over the interval for this collection is {(100 * 125) + (50 * 0.25 * 100) + (500 * 18.8) + (100 * 0.25)} / 171.3 = 128 (1.28%.)

The Energy Accuracy of an aggregated energy value will change over time as the relative contributions of the child devices change.

### 5B-6.3.1.4 Energy Accuracy Basis, Attribute #4

This optional attribute defines the basis of the Energy Accuracy attribute. If this attribute is not present it is assumed to have the default value of zero. It may be one of the values below:

- 0 (default): Accuracy expressed as 100 times the percent of reading
- 1: Accuracy expressed as 100 times the percent of full-scale reading (requires the user or vendor to specify the value of the Full Scale Reading attribute
- 2: Accuracy expressed as 100 times the absolute error expressed in the applicable energy consumption rate (power) units of the associated Energy Object.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-6.3.1.5 Full Scale Reading, Attribute #5

This conditional attribute defines the full scale energy consumption rate (power) reading of a measuring device that supports the Base Energy Object. It is required only when the value of the Energy Accuracy Basis attribute is 1.

### 5B-6.3.1.6 Data Status, Attribute #6

This conditional attribute may be used to indicate if conditions exist that may adversely affect the validity of the data reported by a Base Energy Object. Examples of these conditions include:

- An aggregator is unable to sample data from one or more of its child instances.
- An energy meter is in a state in which metering accuracy is compromised, such as input over-range, power-on self test, etc.
- When aggregating non-electrical energy, the units of energy in one or more child instances do not match that of the associated Non-Electrical Energy Object instance.
- An aggregation path does not point to a correct type of object instance.

The semantics of the Data Status attribute value varies depending upon the value of the Energy Capabilities attribute, as described below.

**Table 5B-6-7 Attribute #6, Values for Data Status**

| Energy Capabilities Attribute Value | Data Status Attribute Semantics of Value |
|---|---|
| 0 = Energy Measured,<br>1 = Energy Derived<br>2 = Energy Proxy | 0 = No Errors<br>1 = Not Metering<br>2 – 65535 Reserved |
| 3 = Energy Aggregated<br>5 = Non-Electrical Aggregated | 0 = No Errors<br>non-zero = Member ID of Aggregation Path EPATH Array that represents the first child object with errors detected (See Aggregation Application Considerations section) |
| 4 = Energy Rate Fixed | 0 = No Errors (only permitted value)<br>1 – 65535 Reserved |

### 5B-6.3.1.7 Consumed Energy Odometer, Attribute #7

The Consumed Energy Odometer represents a range of 0 kWh to 999,999,999,999.999 kWh. The exponent base (n) shall be -3 represented as $10^{-3}$. See the defined constructed data type description of ODOMETER in Appendix C-6.2.3, Defined Constructed Data Types.

The Consumed Energy odometer is a conditional attribute. It shall be required in the following cases:

- The instance does not report only power values using Attribute #10 (Attribute #2 is a value other than 4), and
- The instance reports energy consumption.

When the Consumed Energy Odometer is implemented, the consumed energy value is computed periodically on an incremental basis. During each time period, the incremental consumed energy is added to the Consumed Energy Odometer.

The Consumed Energy Odometer value at a single point in time is of little value. The difference between the odometer values at two or more points in time is of interest, as the difference represents the amount of energy consumed by a process over the interval of time described by the two points. The consumer of this data must be able to discern and correctly interpret data rollover events so as to derive correct interval consumption values.

**Important**: This odometer shall never decrement.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-6.3.1.7.1 Energy Reporting Units (Applies to Attributes #7, #8 and #9)

Attributes 7 - 9 are used to report consumed, generated and total net energy values. Reporting units are normalized in units of kilowatt-hours (kWh). Converting to gigajoules (GJ), to harmonize with energy reporting standards defined by the "Global Reporting Initiative" (GRI)[2], requires multiplying the value in kWh by a factor of 0.0036. The following tables are derived from the GRI G3.1 Guidelines Environmental IP EN3[3] and specifies conversions for energy sources to kilowatt-hours.

| Coal | kWh |
|---|---|
| Tonne (metric) | 7220 |
| Ton (short) | 6550 |
| Ton (long) | 7340 |

| Crude Oil | kWh |
|---|---|
| Barrel | 1730 |
| Tonne (metric) | 12400 |
| Ton (short) | 11300 |
| Ton (long) | 12600 |

| Gasoline | kWh |
|---|---|
| Gallon | 34.7 |
| Tonne (metric) | 12400 |

| Diesel | kWh |
|---|---|
| Gallon | 38.3 |
| Tonne (metric) | 12000 |

| Fuel Oil | kWh |
|---|---|
| Gallon | 40 |
| Tonne (metric) | 11200 |

| Natural Gas | kWh |
|---|---|
| Therm | 29.3 |
| 1000 Cubic Feet (Mcf) | 307 |
| 1000 cubic M | 10800 |
| MMBtu | 293 |

| Electricity | kWh |
|---|---|
| kWh | 1 |
| MWh | 1000 |
| GWh | 10^6 |

2 The Global Reporting Initiative (GRI) is a network-based organization that defines and publishes a sustainability reporting framework. For more information on GRI go to: http://www.globalreporting.org/AboutGRI/

3 G3.1 Guidelines are an update and completion of the third generation of GRI's Sustainability Reporting Guidelines. The Guidelines are the cornerstone of GRI's Reporting Framework which provides guidance on how organizations can disclose their sustainability performance. Guidelines can be downloaded at the following URL: http://www.globalreporting.org/ReportingFramework/G31Guidelines/


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-6.3.1.8 Generated Energy Odometer, Attribute #8

The Generated Energy Odometer represents a range of 0 kWh to 999,999,999,999.999 kWh. The exponent base (n) shall be -3 represented as $10^{-3}$. See the defined constructed data type description of ODOMETER in Appendix C-6.2.3 Defined Constructed Data Types.

The Generated Energy odometer is an optional a conditional attribute. It shall be required in the following cases:

- The instance does not report only power values using Attribute #10 (Attribute #2 is a value other than 4), and
- The instance reports energy generation.

When the Generated Energy Odometer is implemented, the generated energy value is computed periodically on an incremental basis. During each time period, the incremental generated energy is added to the Generated Energy Odometer.

The Generated Energy Odometer value at a single point in time is of little value. The difference between the odometer values at two or more points in time is of interest, as the difference represents the amount of energy generated by a process over the interval of time described by the two points. The consumer of this data must be able to discern and correctly interpret data rollover events so as to derive correct interval generation values.

Important: This odometer shall never decrement.

### 5B-6.3.1.9 Net Energy Odometer, Attribute #9

The Net Energy Odometer represents a range of -999,999,999,999.999 kWh to 999,999,999,999.999 kWh . The exponent base (n) shall be -3 represented as $10^{-3}$. See defined constructed data type description of SIGNED-ODOMETER in Appendix C-6.2.3, Defined Constructed Data Types.

When the Net Energy Odometer attribute is implemented, energy is computed periodically on an incremental basis. During each time period, the difference between the incremental consumed and generated energy values is added to the Net Energy Odometer. A positive value represents net energy consumed. A negative value represents net energy generated.

The Net Energy Odometer value at a single point in time is of little value. The difference between the odometer values at two or more points in time is of interest, as the difference represents the amount of energy consumed or generated by a process over the interval of time described by the two points. The consumer of this data must be able to discern and correctly interpret data rollover events so as to derive correct interval values.

The Net Energy Odometer is an optional attribute.

It is strongly recommended that devices that incorporate the Base Energy Object develop energy values internally within the device, and present such energy values using the Energy Odometer attribute(s).

### 5B-6.3.1.10 Energy Transfer Rate, Attribute #10

This attribute expresses the time rate of energy transfer (e.g. Power) as a REAL value, expressed in engineering units of kilowatts (kW). A positive value indicates consumption of energy, while a negative value indicates generation of energy.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

For devices that report attribute 2 (Base Energy Object Capabilities) values of 0 = Energy Measured, 1 = Energy Derived, or 2 = Energy Proxy, this attribute reports an instantaneous power value. Simple devices without the capability to measure energy are permitted to report an attribute 2 (Base Energy Object Capabilities) value of 4 = Energy Rate Fixed, and report a fixed or bi-state nominal or rated value in this attribute. In this case energy odometer attributes are not required. It is encouraged that energy and energy transfer rate data are based on measured values.

### 5B-6.3.1.11 Energy Transfer Rate User Setting, Attribute #11

This attribute stores a value in kilowatts (kW) which may be set by the user or in the manufacture of the device for use in deriving values of energy consumption (or generation).

If the Energy Capability (attribute #2) value is 0, “Energy Measured”, this attribute is not needed since power and energy are measured directly.

If the Energy Capability (attribute #2) value is 1, “Energy Derived”, this attribute can store a value for use in deriving reported values of Energy Transfer Rate (attribute #10) and the Net Energy Odometer (attribute #9).

If the Energy Capability (attribute #2) value is 2, “Energy Proxy”, or 4, “Energy Rate Fixed”, the device manufacturer shall be responsible to determine and document the method of use of this attribute and how it relates to Energy Transfer Rate (attribute 10) and the Consumed Energy Odometer (attribute #7) or Generated Energy Odometer (attribute #8).

If the Energy Capability (attribute #2) value is 3, “Energy Aggregated”, or 5, “Non-Electrical Aggregated”, this attribute is not needed since power and energy are aggregated from the child devices..

### 5B-6.3.1.12 Energy Type Specific Object Path, Attribute #12

This attribute may contain a path to an Electrical Energy Object instance (class code 0x4F), a path to a Non-Electrical Energy Object instance (class code 0x50) or a Null path (a Path Size value of zero (0)). Pertinent data associated with a given energy type differs from energy type to energy type. Energy type specific data is segregated from this generic Base Energy Object, but can be associated with an instance of the Base Energy Object by using this attribute. The class code and instance number of an Electrical or Non-Electrical object instance are encoded here. For Electrical energy, this attribute can encode an instance of the Electrical Energy Object (class code 0x4F) in this attribute if that object is implemented for this Base Energy Object Instance. Electrical energy consuming/producing devices whose electrical energy values are measured by current and voltage sensors and therefore report a value of 0=Energy Measured in attribute #2 (Base Energy Object Capabilities), are encouraged to implement the Electrical Energy Object and report the various currents, voltages and power levels that are measured in that object. When this is the case, this attribute value would be:

03 00 21 00 4F 00 24 01

Where the first 2 bytes (03 00) indicate a Padded EPATH of size 3 words, followed by the actual Padded EPATH. See Appendix C of this specification for a complete discussion of EPATH encoding.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5B-6.1 Energy Type Specific Object Path Example**

Electrical devices that report a value of 2=Energy Proxy in attribute #2 (Base Energy Object Capabilities) derive an energy value and may not need to implement the Electrical Energy Object because no further pertinent electrical data is available. This is true also with devices that report a value of 4=Energy Rate Fixed in attribute #2, which report a fixed or bi-state electrical power rating. In these cases, this attribute value would be 00 00, indicating no energy type-specific object instance is associated with this instance of the Base Energy Object.

When this object instance is used as an energy aggregator (Attribute #2, Base Energy Object Capabilities is 3) an energy type specific energy object instance may or may not be used. If an instance of the Base Energy Object is being used to aggregate energy type specific objects, an energy type specific object shall not be specified and the value of this attribute shall be zero (0).

When this object instance is used as a non-electrical energy aggregator (attribute #2, Base Energy Object Capabilities = 5) the Energy Type Specific Object Path shall point to a Non-Electrical Energy Object instance. This instance shall define the non-electrical measurement units to be used for the aggregation and be the recipient of the aggregated odometer values of non-electrical energy.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-6.3.1.13 Energy Aggregation Paths, Attributes #13 and #14

The Base Energy Object provides a means of aggregating energy at various levels of an enterprise (in kWh or in native non-electrical units) and presenting this data consistently at all levels; from the device level at the bottom of the energy tree, up through the machine or process level, and through the factory and enterprise level. Consider the following simple system for aggregating kWh:

**Figure 5B-6.2 Energy Aggregation Example System**

In this example, the Machine Controller will aggregate the energy values of the 3 devices whose link addresses are given.

Each device, including the Machine Controller, implements an instance of the Base Energy Object and the Electrical Energy Object. The three devices at the bottom of the energy tree are electrical energy measuring devices, and the Machine Controller implements an instance of the Base Energy Object and the Electrical Energy Object to aggregate the data from the measuring devices. The Machine Controller uses attribute #13 and #14 of the Base Energy Object to encode paths to the Base Energy Object instance in each of the Motor Controllers whose data is to be aggregated. In the above example, the logical encoding of the paths to the Base Energy Objects in the 3 measuring devices that would be used in attribute #14 “Energy Aggregation Paths” in the machine controller would be:

```
0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30 35 00 21 00
4E 00 24 01 0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30
36 00 21 00 4E 00 24 01 0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33
37 2E 31 30 37 00 21 00 4E 00 24 01
```

Note: See Appendix C for a complete discussion of path encoding.

The **bold** portions of the path below (**0C 00**) indicate that the Path Size (in hex words) for each of the 3 encoded paths:

```
0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30 35 00 21 00
4E 00 24 01 0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30
36 00 21 00 4E 00 24 01 0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33
37 2E 31 30 37 00 21 00 4E 00 24 01
```

The **bold** portions of the path below (**12**) indicate that the segment type is Port and that it is a multi-byte address for TCP port 2:

```
0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30 35 00 21 00
4E 00 24 01 0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30
36 00 21 00 4E 00 24 01 0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33
37 2E 31 30 37 00 21 00 4E 00 24 01
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The **bold** portions of the path below (**0F**) indicate that the encoded link address that follows is 15 bytes long:

```
0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30 35 00 21 00
4E 00 24 01 0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30
36 00 21 00 4E 00 24 01 0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33
37 2E 31 30 37 00 21 00 4E 00 24 01
```

The **bold** portions of the path below are the encoded link addresses to the 3 energy measuring devices (130.151.137.10x):

```
0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30 35 00 21 00
4E 00 24 01 0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30
36 00 21 00 4E 00 24 01 0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33
37 2E 31 30 37 00 21 00 4E 00 24 01
```

The **bold** portions of the path below are the encoded class and instance of the Base Energy Object for each energy-measuring device:

```
0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30 35 00 21 00
4E 00 24 01 0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30
36 00 21 00 4E 00 24 01 0C 00 12 0F 31 33 30 2E 31 35 31 2E 31 33
37 2E 31 30 37 00 21 00 4E 00 24 01
```

It is worth noting that symbolic encoding could also be used. The type of path encoding that is used by the aggregator is the decision of the application or aggregator device developer. See Appendix C of this specification for a complete discussion of path encoding.

The following diagram shows the various object relationships that would be defined in this example.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5B-6.3 Example Energy Aggregation Object Relationships**

**Machine Controller Base Energy Object Instance 1**

| | |
|---|---|
| 1) Energy/Resource Type | 1 = Electrical |
| 2 Energy Object Capabilities | 3 = Energy Aggregated |
| 3 Energy Accuracy | Aggregated Accuracy Value |
| 4 Energy Accuracy Basis | |
| 5 Full Scale Reading | |
| 6 Data Status | |
| 7 Consumed Energy Odometer | Aggregated Energy Value |
| 8 Generated Energy Odometer | Aggregated Energy Value |
| 9 Net Energy Odometer | Aggregated Energy Value |
| 10 Energy Transfer Rate | Aggregated Value |
| 11 Energy Transfer Rate User Setting | Not used by aggregator |
| 12 Energy Type Specific Object Path | 00 00 |
| 13 Energy Aggregation Path Array Size | 3 |
| 14 Energy Aggregation Paths | 0C 00 12 0F 31 33 30 2E 31 35 31 2E 31<br>33 37 2E 31 30 35 00 21 00 4E 00 24 01<br>0C 00 12 0F 31 33 30 2E 31 35 31 2E 31<br>33 37 2E 31 30 36 00 21 00 4E 00 24 01<br>0C 00 12 0F 31 33 30 2E 31 35 31 2E 31<br>33 37 2E 31 30 37 00 21 00 4E 00 24 01 |

| |
|---|
| 1) Energy/Resource Type |
| 2 Energy Object Capabilities |
| 3 Energy Accuracy |
| 4 Energy Accuracy Basis |
| 5 Full Scale Reading |
| 6 Data Status |
| 7 Consumed Energy Odometer |
| 8 Generated Energy Odometer |
| 17) Metering State |

**Motor Starter Base Energy Object Instance 1**

| |
|---|
| 1) Energy/Resource Type |
| 2 Energy Object Capabilities |
| 3 Energy Accuracy |
| 4 Energy Accuracy Basis |
| 5 Full Scale Reading |
| 6 Data Status |
| 7 Consumed Energy Odometer |
| 8 Generated Energy Odometer |
| 17) Metering State |

**Circuit Breaker Base Energy Object Instance 1**

| |
|---|
| 1) Energy/Resource Type |
| 2 Energy Object Capabilities |
| 3 Energy Accuracy |
| 4 Energy Accuracy Basis |
| 5 Full Scale Reading |
| 6 Data Status |
| 7 Consumed Energy Odometer |
| 8 Generated Energy Odometer |
| 17) Metering State |

**DC Power Supply Base Energy Object Instance 1**


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Consider the following simple system for natively aggregating cubic feet of natural gas:

**Figure 5B-6.4 Non-Electrical Energy Aggregation Example System**

In this example, the Controller will aggregate the energy values of the 3 Furnaces. Each Furnace, including the Controller, implements an instance of the Base Energy Object and the Non-Electrical Energy Object. The three Furnaces at the bottom of the energy tree are non-electrical energy measuring devices. The Non-Electrical Object instance associated to the Base Energy Object instance in the Controller defines the units and conversion factors of the energy to be aggregated. Each Non-Electrical Energy Object instance in a Furnace must have the same units and conversion factors in order to be aggregated correctly. The Controller uses attribute 13 and 14 of the Base Energy Object instance to encode paths to the Non-Electrical Energy Object instance in each of the Furnaces whose data is to be aggregated. In the above example, the logical encoding of the paths to the Non-Electrical Energy Objects in the 3 Furnaces that would be used in attribute 14 “Energy Aggregation Paths” in the Controller would be:

```
0C 00 12 0F 31 39 32 2E 31 36 38 2E 31 30 30 2E 31 30 31 00 21 00
50 00 24 01 0C 00 12 0F 31 39 32 2E 31 36 38 2E 31 30 30 2E 31 30
32 00 21 00 50 00 24 01 0C 00 12 0F 31 39 32 2E 31 36 38 2E 31 30
30 2E 31 30 33 00 21 00 50 00 24 01
```

Note: See Appendix C for a complete discussion of path encoding.

Note: Unlike aggregation in kWh, when aggregating non-electrical energy, the path references the Non-Electrical Energy Object instance rather than a Base Energy Object instance.

Refer to the previous example to explore decoding of the paths.

The following diagram shows the various object relationships that would be defined in this example.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5B-6.5 Example Non-Electrical Energy Aggregation Object Relationships**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-6.3.1.14 Odometer Reset Enable, Attribute #16

The ”Odometer Reset Enable” attribute determines if energy odometer attributes are affected by the reset service. This parameter has a default value of 0 = disabled. If this attribute is not implemented, then odometer resetting will be disabled. If this attribute is set to the value 1 = enabled, then the Base Energy Object reset service will reset all energy odometers associated with the specified instance of the Base Energy Object, including any energy odometers in associated energy type specific objects specified in attribute 12 “Energy Type Specific Object Path”. The resetting of all energy odometers associated with an instance of the Base Energy Object shall be an atomic operation.

### 5B-6.3.1.15 Metering State, Attribute #17

The “Metering State” attribute reflects the metering state described in section 5B-6.6, Behavior. A value of zero (0) signifies that the Base Energy Object instance is in the “Not Metering” state. A value of one (1) signifies that the Base Energy Object instance is in the “Metering” state. Note that the metering state of an instance is non-volatile.

### 5B-6.3.1.16 Extended Data Status, Attribute #18

This conditional attribute is an enumeration that provides additional information related to the value of the Data Status, Attribute #6, as shown below.

**Table 5B-6-8 Attribute #18, Extended Data Status**

| Value | Extended Data Status Semantics of Value |
|---|---|
| 0 | No errors |
| 1 | No associated Non-Electrical Energy Object instance |
| 2 | Units of child Non-Electrical Energy Object instance do not match associated Non-Electrical Energy Object instance |
| 3 - 65535 | Reserved, shall be ignored |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-6.4 Common Services

**Table 5B-6-9 Base Energy Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 01hex | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this objects attributes. |
| 03hex | Optional | Optional | Get_Attribute_List | Returns the contents of the selected gettable attributes. |
| 04hex | n/a | Optional | Set_Attribute_List | Modifies the contents of the selected settable attributes. |
| 05hex | Optional[1] | Conditional[2] | Reset | Invokes the reset service for a Base Energy Object instance |
| 08hex | Optional | n/a | Create | Instantiate a new instance of the Base Energy Object |
| 09hex | n/a | Optional | Delete | Delete an instance of the Base Energy Object |
| 0Ehex | Required | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 10hex | n/a | Conditional[3] | Set_Attribute_Single | Modifies an attribute value. |
| 18hex | n/a | Optional | Get_Member | Used to get individual device EPATHS in instance attribute #14 |
| 19hex | n/a | Optional | Set_Member | Used to set individual device EPATHS in instance attribute #14 |
| 1Ahex | n/a | Optional | Insert_Member | Used to insert an individual device EPATH into the array in attribute #14 and increment attribute #13 |
| 1Bhex | n/a | Optional | Remove_Member | Used to remove an individual device EPATH from the array in attribute #14 and decrement attribute #13 |

Table Footnotes:

1. Invoking the Reset service at the Class level resets the energy data across all Base Energy Object instances for a device, including each associated energy type-specific object that exists, but not including child devices linked to an aggregator.
2. Required if Attribute #16 (Odometer Reset Enable) is implemented.
3. Required if any of the optional Set attributes are implemented.

See Appendix A for the definition of these services

### 5B-6.4.1 Reset Service

When an instance of the Base Energy Object receives a Reset request, it:

- determines if it can provide the type of reset requested
- responds to the request
- attempts to perform the type of reset requested.

If instance attribute 16 (Odometer Reset Enable) is not supported, then a type 0 and type 1 reset service to the Base Energy Object returns the value 08 = Service Not Supported in the General Status Code field, and the reset type (0 or 1) in the Extended Code field

If instance attribute 16 (Odometer Reset Enable) is implemented and has a value of 0=disable, then type 0 and type 1 reset services to the Base Energy Object return the value 10 hex = Device State Conflict in the General Status Code field, and the reset type (0 or 1) in the Extended Code field.

The Reset common service has the following object–specific parameter

**Table 5B-6-10 Reset Service Parameter**

| Name | Data Type | Description of Request Parameter | Semantics of Value |
|---|---|---|---|
| Reset Type | USINT | Type of Reset | See table below |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The parameter for the Reset common service has the following specifications:

**Table 5B-6-11 Reset Service Parameter Values**

| Value | |
|---|---|
| 0 | Reset the Consumed, Generated and Net Energy Odometers (attributes 7, 8 and 9) to the value zero (0)[1] |
| 1 | Reset the Consumed, Generated and Net Energy Odometers (attributes 7, 8 and 9) to the value zero (0) and return the Base Energy Object as closely as possible to the factory default configuration[1] |
| 2 | Return the Base Energy Object as closely as possible to the factory default configuration without resetting Energy Odometers (attributes 7, 8 and 9). |
| 3-99 | Reserved by ODVA |
| 100-199 | Vendor Specific |
| 200-255 | Reserved by ODVA |

Table Footnotes

1 The use of the Type 0 and Type 1 services which reset the energy odometers should be discouraged, because it can cause an unexpected change in odometer values that may lead to misinterpretation of data by aggregators or other data clients resulting in errors.

### 5B-6.4.2 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5B-6-12 Get_Attributes_All Response – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 [1] |
| 2 | UINT | Max Instance | 1 |
| 3 | UINT | Number of Instances | 1 |
| 6 | UINT | Max ID Number of Class Attributes | 0 |
| 7 | UINT | Max ID Number of Instance Attributes | 0 |

Table Footnotes:

1. The default value only applies for Revision 1 of this object. Newer revisions require that this attribute be implemented, which will override this default value.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-6-13 Get_Attributes_All Response – Instance Level – Revision 01 and 02**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Energy/Resource Type | |
| 2 | UINT | Energy Object Capabilities | |
| 3 | UINT | Energy Accuracy | |
| 4 | UINT | Energy Accuracy Basis | 0 |
| 5 | REAL | Full Scale Reading | 0.0 |
| 6 | UINT | Data Status | 0 |
| 7 | ODOMETER | Consumed Energy Odometer | {0.0.0.0.0} |
| 8 | ODOMETER | Generated Energy Odometer | {0.0.0.0.0} |
| 9 | SIGNED_ODOMETER | Net Energy Odometer | {0.0.0.0.0} |
| 10 | REAL | Energy Transfer Rate | 0.0 |
| 11 | REAL | Energy Transfer Rate User Setting | 0.0 |
| 12 | STRUCT of UINT, Padded EPATH [1] | Energy Type Specific Object Path | (UINT) 0 |
| 13 | UINT | Energy Aggregation Path Array Size | 0 |
| 14 | Array of STRUCT of UINT, Padded EPATH [1] | Energy Aggregation Paths | Null |
| 15 | STRINGI | Energy Identifier | Number = 0 |
| 16 | BOOL | Odometer Reset Enable | 0 |
| 17 | BOOL | Metering State | 1 |

Table Footnotes:

1. For revision 2 of the Base Energy Object, the UINT (Path Size) is specified in words. For revision 1 of object, the UINT (Path Size) is specified in bytes.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

### 5B-6.4.3 Create Service

Creates a new instance of the Base Energy Object in a device that supports dynamic instances. Examples of such devices are controllers and Modbus translators. Instance numbers are allocated starting from 1. The smallest available instance number greater than 0 is allocated. This service does not need to be supported in devices that implement only static instances of the Base Energy Object. Table 5B-6.14 defines the request parameters for the Create service.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-6-14 Create Service Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Energy Resource Type | UINT | Specifies the type of energy managed by this energy instance. This value will be reported in attribute 1 of the created instance. | 0 = Generic<br>1 = Electricity<br>2 = Non-Electrical<br><br>See section 5B-6.3.1.1 |
| Base Energy Object Capabilities | UINT | The Energy Capabilities of the Device represented by the created Base Energy Object instance. This value will be reported in attribute 2 of the created instance. | Range: 0-4.<br>See section 5B-6.3.1.2 |
| Energy Accuracy | UINT | Specifies the accuracy of power and energy values. This value will be reported in attribute 3 of the created instance | 0 = unknown or the created instance will report a measured or aggregated energy accuracy value. |
| Create Energy Type Specific Object Class Code | UINT | Class code of the energy type specific object instance to be created and linked to the Base Energy Object. If none is to be created, this value is zero (0) | Class code of the Electrical Energy Object (class code 0x4F), Non-Electrical Object (class code 0x50) or no energy type specific object (value of 0000) This class code is used to build the "Energy Type Specific Object Path" in Attribute 12 of the created object. |
| Optional Vendor Specific attribute data | Struct of<br>UINT (attribute #)<br>UINT (data size)<br>Vendor Specifiic attribute data | Any number of vendor specific attribute data can be passed to the create service for the purpose of initializing vendor specific "Get" attributes | These parameters are Optional |

Table 5B-6.15 defines the response parameters for the Create service.

**Table 5B-6-15 Create Service Response Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Instance Number Created | UINT | Instance number allocated | 1 - 65535 |

### 5B-6.4.4 Delete Service

The Delete service is used to de-allocate all resources for a specified instance of the Base Energy Object, including the Electrical or Non-Electrical Energy object instance pointed to in attribute 12 "Energy Type Specific Object Path". The Delete service shall not be supported at the class level. This service does not need to be supported in devices that implement static instances of the Base Energy Object.

There are no request parameters for the Delete service.

There are no response parameters for the Delete service.

When the Delete service references the Instance Number of a static Base Energy Object instance that cannot be deleted, the error code 2D hex = Instance Not Deletable will be returned in the General Error Code, and the Instance Number of the static Base Energy Object instance will be returned in the Extended Error Code.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-6.5 Object Specific Services

**Table 5B-6-16 Base Energy Object Specific Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| $4B_{hex}$ | Conditional [1] | Conditional[2] | Start_Metering | Starts the metering of power and energy for a device (class level) or an instance. |
| $4C_{hex}$ | Conditional [1] | Conditional[2] | Stop_Metering | Stops the metering of power and energy for a device (class level) or an instance. |

Table Footnotes:
1 These services are optional as a pair. If one of these services is supported at the class level, the other must also be supported.
2 These services are optional as a pair. If one of these services is supported at the instance level, the other must also be supported.

### 5B-6.5.1 Start_Metering Service

This service provides a mechanism to programmatically start the metering process. If this service is not implemented in a device, energy metering begins on power up and continues as long as the device is operational. When this service is invoked at the class level, energy metering is enabled for the entire device i.e. for all instances the Base Energy Object, including its associated energy type-specific object if one exists (but not including child devices linked to an aggregator). When this service is invoked at the instance level, energy metering is enabled for the specified Base Energy Object instance, including its associated energy type-specific object if one exists.

The Start_Metering service has no request parameters.

### 5B-6.5.2 Stop_Metering Service

This service provides a mechanism to programmatically stop the metering process. If this service is not implemented in a device, energy metering cannot be turned off. When this service is invoked at the class level, energy metering is stopped for the entire device i.e. for all instances of the Base Energy Object, including its associated energy type-specific object if one exists (but not including child devices linked to an aggregator). When this service is invoked at the instance level, energy metering is stopped for just the specified Base Energy Object instance, including its associated energy type-specific object if one exists.

The Stop_Metering service has no request parameters.

## 5B-6.6 Behavior

The behavior of static Base Energy Object instances is illustrated in Table 5B-6.17 below. Note that if the Start_Metering and Stop_Metering services are not supported, the instance enters and remains in the Metering state. If the Start_Metering and Stop_Metering services are supported, the Metering State of the device is preserved through power cycles. The default value of Metering State is 1 = Metering.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5B-6.6 Static Base Energy Object State Diagram**

**Table 5B-6-17 Base Energy Object State Event Matrix**

| Event | State | |
|---|---|---|
| | **Not Metering** | **Metering** |
| Start_Metering | Transition to Metering | Ignore event |
| Stop_Metering | Ignore Event | Transition to Not Metering |
| Type 0 Reset | Reset all Energy Odometers to 0 for the Base Energy Object and Energy Type Specific Object pointed to by the Energy Type-Specific Object Path (attribute 12). | |
| Type 1 Reset | Reset Base Energy Object and Energy Type-Specific Object pointed to by the Energy Type-Specific Object Path (attribute 12) to factory defaults | |
| Type 2 Reset | Reset the configuration attributes of the Base Energy Object and Energy Type-Specific Object pointed to by the Energy Type-Specific Object Path (attribute 12) without resetting Energy Odometers | |
| Get_Attributes_All | Validate and service the request and send the appropriate response | |
| Get_Attribute_List | | |
| Set_Attribute_List | | |
| Get_Attribute_Single | | |
| Set_Attribute_Single | | |

The behavior of dynamic Base Energy Object instances is illustrated in Table 5B-6.18 below. Note that if the Start_Metering and Stop_Metering services are not supported, the device enters and remains in the Metering state upon creation. If the Start_Metering and Stop_Metering services are supported, the Metering State of the instance is preserved through power cycles.

**Figure 5B-6.7 Dynamic Base Energy Object State Diagram**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-6-18 Dynamic Base Energy Object State Event Matrix**

| Event | State | | |
|---|---|---|---|
| | **Non-Existent** | **Not Metering** | **Metering** |
| Create | Class instantiates a Base Energy Object. Transition to Metering | Not applicable | Not applicable |
| Delete | Error: Object does not exist (General Error Code 0x16) | Release all associated resources. Transition to Non-Existent | |
| Start_Metering | Error: Object does not exist (General Error Code 0x16) | Transition to Metering | Ignore event |
| Stop_Metering | Error: Object does not exist (General Error Code 0x16) | Ignore Event | Transition to Not Metering |
| Type 0 Reset | Error: Object does not exist (General Error Code 0x16) | Reset all Energy Odometers to 0 for the Base Energy Object and Energy Type Specific Object pointed to by the Energy Type-Specific Object Path (attribute 12) | |
| Type 1 Reset | Error: Object does not exist (General Error Code 0x16) | Reset Base Energy Object and Energy Type-Specific Object pointed to by the Energy Type-Specific Object Path (attribute 12) to factory defaults | |
| Type 2 Reset | Error: Object does not exist (General Error Code 0x16) | Reset the configuration attributes of the Base Energy Object and Energy Type-Specific Object pointed to by the Energy Type-Specific Object Path (attribute 12) without resetting Energy Odometers | |
| Get_Attributes_All | Error: Object does not exist (General Error Code 0x16) | Validate and service the request and send the appropriate response | |
| Get_Attribute_List | | | |
| Set_Attribute_List | | | |
| Get_Attribute_Single | | | |
| Set_Attribute_Single | | | |

## 5B-6.7 Aggregation Application Considerations

An aggregator is a CIP Energy aware device that contains an instance of the Base Energy Object whose Base Energy Object Capabilities attribute value is 3, “Energy Aggregated” or 5, “Non-Electrical Aggregated”. This Base Energy Object instance (“aggregator instance”) acts as an independent meter. Its odometers accumulate incremental changes in energy values that are sampled from members of a collection of child Base Energy Object or Non-Electrical Energy Object instances. The aggregator instance odometer values may have no direct relationship with the odometer values in the child instances. Several events alter the relative odometer values and require special handling. Refer to Item 4 in the list below.

An aggregator will typically be a somewhat sophisticated device such as a controller or software application. It needs to have an internal clock, which may under certain circumstances need to be synchronized to absolute time (local or UTC). An aggregator is both a client to its “child” instances (the devices whose energy values are being aggregated) and a server to information applications higher in the architecture.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Child instances may report energy as odometer values or as bi-state (on, off) power (rate of energy consumption) values. Three types of odometer values exist. Two monotonic odometers represent energy consumption (load) and energy generation (generator). The net energy odometer is bi-directional. The net odometer increases or decreases its value as its associated energy resource consumes or generates energy respectively. Its value accumulates the difference between the incremental consumed and generated energy values.

Consumed and generated energy odometers roll over to zero when they reach their maximum value. Net energy odometers roll over to zero when they reach their maximum positive or negative values. Because the range of the ODOMETER and SIGNED-ODOMETER data types is very large, (e.g. +/- 999.9 TWh), a rollover is a rare event.

Energy aware devices typically update their odometer values at regular intervals. Energy values are the time integral of power values. To maintain accuracy, the update interval should be small, in the range of milliseconds to seconds. Each interval, a device adds the incremental consumed energy to the consumed energy odometer, and adds incremental generated energy to the generated energy odometer. Then the difference between the incremental consumed and generated energy during the interval is added to the net energy odometer. Rollovers of each odometer occur independently of each other.

As stated in the Semantics section above, the value of an energy odometer at a single point in time is of little or no value. The value of energy consumed or produced over an interval of time (the interval of interest) is the difference between the odometer value sampled at the end of the interval and its value sampled at the beginning of the interval (an aggregator must sample its child instances twice in order to derive its first aggregated energy odometer value). If a rollover occurs during the interval of interest, the client or aggregator must correctly account for it. The interval of interest is typically much smaller than the interval between rollovers, e.g. minutes, hours or days compared with years. To satisfy Nyquist's theorem, the sampling interval must be no more than half the minimum rollover interval.

Let's consider aggregators of child instances that all report odometer style data. Aggregators sample the odometer data of their child instances periodically and develop aggregated energy odometers to present to the aggregator's clients. During each interval, the aggregator, for each child instance:

1. Samples the present Consumed Energy Odometer and Generated Energy Odometer of each child instance.
2. Computes the differences between the present values and the previously buffered values of each odometer to obtain the child instance's incremental consumed and/or generated energy.
3. Buffers the present values of each child instance's odometers for use in the next interval.
4. Examines each instance's incremental, consumed and/or generated energy value to determine if an event occurred that requires special handling. Examples of these events include:
    a. A child instance's odometer rollover occurred during the interval (a positive or negative difference approaching in magnitude the rollover value of the child instance's odometer).
    b. An unexpected change in the child instance's odometer value, caused, for instance, by a device replacement or reset.
    c. A child instance not responding to a sample request from the aggregator.
    d. A child instance, which had been nonresponsive for a time, again responding to a sample request.
    e. A child instance that was deleted.
    f. A child instance stops metering.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

5. If needed, makes an appropriate correction to the difference values. Examples of corrections include:
   a. For example 4.a. add or subtract the rollover value (10^12 units) as needed.
   b. For example 4.b or 4.d, take a second sample of the child instance's odometer value after a short wait (less than the normal sampling interval but greater than the data refresh interval of the child instance), and then extrapolate the difference to the normal sampling interval
   c. For examples 4.b., 4.c, 4.e. or 4.f, write the Aggregation Path EPATH Array Index value that represents the first child instance with errors detected
6. Adds the child instance's corrected incremental consumed energy to the appropriate odometer(s):
   a. If the aggregator instance's Capabilities attribute is 3, "Energy Aggregated", adds the incremental consumed energy to its Consumed Energy Odometer if it exists.
   b. If the aggregator instance's Capabilities attribute is 5, "Non-Electrical Aggregated", adds the incremental consumed energy to the Energy Consumed Odometer (if it exists) of the Non-Electrical Energy Object instance referenced by the Energy Type Specific Object Path. Then, converts the Energy Consumed Odometer value to kWh by using the Normalization Constant Multiplier and Normalization Constant Divisor from the referenced Non-Electrical Energy Object instance and stores the converted value in the aggregator instance's Consumed Energy Odometer if it exists.
7. Adds the child instance's corrected incremental generated energy to the appropriate odometer(s):
   a. If the aggregator instance's Capabilities attribute is 3, "Energy Aggregated", adds the incremental generated energy to its Generated Energy Odometer if it exists.
   b. If the aggregator instance's Capabilities attribute is 5, "Non-Electrical Aggregated", adds the incremental generated energy to the Energy Generated Odometer (if it exists) of the Non-Electrical Energy Object instance referenced by the Energy Type Specific Object Path. Then, converts the Energy Generated Odometer value to kWh by using the Normalization Constant Multiplier and Normalization Constant Divisor from the referenced Non-Electrical Energy Object instance and stores the converted value in the aggregator instance's Generated Energy Odometer if it exists.
8. Calculates the child instance's incremental net energy by subtracting the child instance's corrected incremental generated energy from its corrected incremental consumed energy, and then:
   a. If the aggregator instance's Capabilities attribute is 3, "Energy Aggregated", adds the incremental net energy to its Net Energy Odometer if it exists.
   b. If the aggregator instance's Capabilities attribute is 5, "Non-Electrical Aggregated", adds the incremental net energy to the Energy Net Odometer (if it exists) of the Non-Electrical Energy Object instance referenced by the Energy Type Specific Object Path. Then, converts the Energy Net Odometer value to kWh by using the Normalization Constant Multiplier and Normalization Constant Divisor from the referenced Non-Electrical Energy Object instance and stores the converted value in the aggregator instance's Net Energy Odometer if it exists.
9. Takes appropriate action if an aggregated energy odometer's value exceeds its rollover value


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

An aggregator that also aggregates child Base Energy Object instances that report fixed (bi-state) power data would need to, for each such child instance:

1. Recognize the child data as power data
2. Sample the present data from the child
3. Multiply the value of the data by the length of the sampling interval in hours (convert kW to kWh)
4. Add / subtract the incremental energy value to the aggregated consumed, generated and net odometers
5. Takes appropriate action if an aggregated energy odometer's value exceeds its rollover value

## 5B-6.8 Modbus Integration Considerations (Odometers/Base Energy Object in a Modbus Translator) (Informative)

In order to take advantage of energy measurements that already exist in Modbus devices, support for the Base Energy Object may need to be located in the Modbus translator (as described in Volume 7 of the CIP Networks Library).

Support for the Base Energy Object within the Modbus translator has the following goals. To use unmodified Modbus devices is the primary goal. It is desired to acquire energy data from a Modbus device without having to customize the client or the server. If the Modbus device has energy data in locations other than holding registers, then it is the responsibility of the device to place that data in holding registers. There needs to be a simple way to inform the client about Power vs Energy, sampling rates, and Energy related registers addresses, in addition to a messaging strategy.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5B-7 Electrical Energy Object

## Class Code: 4F Hex

This object is used to provide unified electrical energy reporting capability for CIP enabled devices and processes. Energy reporting is typically related to the measurement of a variety of metering results. This object provides for the consistent reporting of electrical energy data. Electrical energy is assigned a separate object to accommodate its alternating and polyphase characteristics, which result in a collection of attributes that are unique among energy sources.

Each instance of the Electrical Energy Object shall be associated with an instance of the Base Energy Object (class code 0x4E). Instance attribute 41, “Associated Base Energy Object Path” is used to define this association.

## 5B-7.1 Revision History

Since the initial release of this object class definition, changes have been made that require a revision update of this object class. The table below represents the revision history:

**Table 5B-7-1 Electric Energy Object Revision History**

| Revision | Reason for object definition change |
|---|---|
| 01 | Initial definition at first release of specification |
| 02 | • Changed data types of energy odometer attributes 1 through 8 to use the newly defined SIGNED_ODOMETER and ODOMETER data types.<br>• Redefined path size in instance attribute 41 to be in WORDs instead of Bytes.<br>• Changed the Need In Implementation property of energy odometer attributes 1 through 3 |

## 5B-7.2 Class Attributes

**Table 5B-7-2 Electrical Energy Object Class Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Required | Get | NV | Revision | UINT | Revision of this object | The current value assigned to this attribute is two (2) |
| 2 | Conditional[1] | Get | NV | Max Instance | UINT | Maximum instance number of an object currently created in this class level of the device | The largest instance number of a created object at this class hierarchy level. |
| 3 | Conditional[1] | Get | NV | Number of Instances | UINT | Number of object instances currently created at this class level of the device. | The number of object instances at this class hierarchy level. |
| 4 thru 7 | These class attributes are described in Volume 1, Chapter 4. Attributes 4 through 7 are optional. | | | | | | |

Table Footnotes:
1 Attributes 2 and 3 are required if the Create and Delete services are supported, or if more than one instance of the Electrical Energy Object is supported


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-7.3 Instance Attributes

**Table 5B-7-3 Electrical Energy Object Instance Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Conditional [1] | Get | NV | Real Energy Consumed Odometer | ODOMETER | The total real energy consumed. | See section 5B-7.3.1.1 |
| 2 | Conditional [2] | Get | NV | Real Energy Generated Odometer | ODOMETER | The total real energy generated. | See section 5B-7.3.1.2 |
| 3 | Optional | Get | NV | Real Energy Net Odometer | SIGNED_ ODOMETER | The running total of Real Energy Consumed minus Real Energy Generated | See section 5B-7.3.1.3 |
| 4 | Optional | Get | NV | Reactive Energy Consumed Odometer | ODOMETER | The total reactive power consumed. | See section 5B-7.3.1.4 |
| 5 | Optional | Get | NV | Reactive Energy Generated Odometer | ODOMETER | The total reactive power generated. | See section 5B-7.3.1.5 |
| 6 | Optional | Get | NV | Reactive Energy Net Odometer | SIGNED_ ODOMETER | The running total of Reactive Energy Consumed minus Reactive Energy Generated | See section 5B-7.3.1.6 |
| 7 | Optional | Get | NV | Apparent Energy Odometer | ODOMETER | The total apparent energy consumed. | See section 5B-7.3.1.7 |
| 8 | Optional | Get | NV | Kiloampere-Hours Odometer | ODOMETER | Accumulated current hours. | See section 5B-7.3.1.8 |
| 9 | Optional | Get | V | Line Frequency | REAL | Line Frequency in Hertz | Hz |
| 10 | Optional | Get | V | L1 Current | REAL | RMS line current in L1. | Amps (A) |
| 11 | Optional | Get | V | L2 Current | REAL | RMS line current in L2. | Amps (A) |
| 12 | Optional | Get | V | L3 Current | REAL | RMS line current in L3. | Amps (A) |
| 13 | Optional | Get | V | Average Current | REAL | RMS line current of three-phase average. | Amps (A) |
| 14 | Optional | Get | V | Percent Current Unbalance | REAL | Percentage Current deviation between phases. | Percent |
| 15 | Optional | Get | V | L1-N Voltage | REAL | RMS line to neutral voltage of L1. | Volts (V) |
| 16 | Optional | Get | V | L2-N Voltage | REAL | RMS line to neutral voltage of L2. | Volts (V) |
| 17 | Optional | Get | V | L3-N Voltage | REAL | RMS line to neutral voltage of L3. | Volts (V) |
| 18 | Optional | Get | V | Average L-N Voltage | REAL | RMS line to neutral voltage of three-phase average. | Volts (V) |
| 19 | Optional | Get | V | L1-L2 Voltage | REAL | RMS L1 to L2 voltage. | Volts (V) |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Electrical Energy Object, Class Code: $4F_{Hex}$

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 20 | Optional | Get | V | L2-L3 Voltage | REAL | RMS L2 to L3 voltage. | Volts (V) |
| 21 | Optional | Get | V | L3-L1 Voltage | REAL | RMS L3 to L1 voltage. | Volts (V) |
| 22 | Optional | Get | V | Average L-L Voltage | REAL | RMS line to line voltage three-phase average. | Volts (V) |
| 23 | Optional | Get | V | Percent Voltage Unbalance | REAL | Percentage Voltage deviation between phases | Percent |
| 24 | Optional | Get | V | L1 Real Power | REAL | L1 real power, signed to show direction. | Watts (W) |
| 25 | Optional | Get | V | L2 Real Power | REAL | L2 real power, signed to show direction. | Watts (W) |
| 26 | Optional | Get | V | L3 Real Power | REAL | L3 real power, signed to show direction. | Watts (W) |
| 27 | Conditional | Get | V | Total Real Power | REAL | Total real power, signed to show direction. | Watts (W) |
| 28 | Optional | Get | V | L1 Reactive Power | REAL | L1 reactive power, signed to show direction. | Volt-amps reactive (VAR) |
| 29 | Optional | Get | V | L2 Reactive Power | REAL | L2 reactive power, signed to show direction. | Volt-amps reactive (VAR) |
| 30 | Optional | Get | V | L3 Reactive Power | REAL | L3 reactive power, signed to show direction. | Volt-amps reactive (VAR) |
| 31 | Optional | Get | V | Total Reactive Power | REAL | Total reactive power, signed to show direction. | Volt-amps reactive (VAR) |
| 32 | Optional | Get | V | L1 Apparent Power | REAL | L1 apparent power. | Volt-amps (VA) |
| 33 | Optional | Get | V | L2 Apparent Power | REAL | L2 apparent power. | Volt-amps (VA) |
| 34 | Optional | Get | V | L3 Apparent Power | REAL | L3 apparent power. | Volt-amps (VA) |
| 35 | Optional | Get | V | Total Apparent Power | REAL | Total apparent power. | Volt-amps (VA) |
| 36 | Optional | Get | V | L1 True Power Factor | REAL | L1 Ratio between power and apparent power.<br>+ = Lead<br>- = Lag | Percent |
| 37 | Optional | Get | V | L2 True Power Factor | REAL | L2 Ratio between power and apparent power.<br>+ = Lead<br>- = Lag | Percent |
| 38 | Optional | Get | V | L3 True Power Factor | REAL | L3 Ratio between power and apparent power.<br>+ = Lead<br>- = Lag | Percent |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Electrical Energy Object, Class Code: $4F_{Hex}$**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 39 | Optional | Get | V | Three Phase True Power Factor | REAL | Ratio between power and apparent power.<br>+ = Lead<br>- = Lag | Percent |
| 40 | Optional | Get | V | Phase Rotation | UINT | The phase rotation of a three-phase system. | 0 = None<br>1 = ABC<br>2 = ACB |
| 41 | Required | Get | NV | Associated Base Energy Object Path | SRUCT of: | Path to associated Base Energy Object instance | See section 5B-7.3.1.10 |
| | | | | Path Size | UINT | Size of path (in words) | |
| | | | | Path | Padded EPATH | See Appendix C for format of this field. | |

Table Footnotes:

1. Attribute 1 is required when Attribute 2 in the Base Energy Object instance that is pointed to by the Associated Energy Object Path attribute (attribute 41) is a value other than 4, “Energy Value Fixed” and the instance reports consumed energy.
2. Attribute 2 is required when Attribute 2 in the Base Energy Object instance that is pointed to by the Associated Energy Object Path attribute (attribute 41) is a value other than 4, “Energy Value Fixed” and the instance reports generated energy.
3. Attribute 27 is required only when Attribute 2 in the Base Energy Object instance that is pointed to by the Associated Energy Object Path attribute (attribute 41) is set to a value of 4, “Energy Value Fixed.”

### 5B-7.3.1 Semantics

Attributes 1-8 all have a data type of ODOMETER or SIGNED_ODOMETER. The exponent base (n) shall be -3 represented as $10^{-3}$. See defined constructed data type description of ODOMETER and SIGNED_ODOMETER in Appendix C-6.2.3, Defined Constructed Data Types.

The units associated with each odometer attribute will be given in the individual attribute semantics sections that follow.

When the energy odometers described in the following sections are implemented, energy values are computed periodically on an incremental basis. During each time period, the incremental consumed real and reactive energies are added to the Real Energy Consumed and Reactive Energy Consumed odometers, the incremental real and reactive generated energies are added to the Real Energy Generated and Reactive Energy Generated odometers, the difference between the incremental consumed and generated real energy is added to the Real Energy Net odometer, and the difference between the incremental consumed and generated reactive energy is added to the Reactive Energy Net odometer.

Note that an energy odometer value at a single point in time is of little value. The difference between the odometer values at two or more points in time is of interest, as the difference represents the amount of energy consumed or generated by a process over the interval of time described by the two points. The consumer of energy odometer data must be able to discern and correctly interpret data rollover events so as to derive correct interval energy values.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5B-7.3.1.1 Real Energy Consumed Odometer, Attribute #1

The Real Energy Consumed Odometer represents a range of 0 kWh to 999,999,999,999.999 kWh. The exponent base (n) shall be -3 represented as $10^{-3}$. See the defined constructed data type description of ODOMETER in Appendix C-6.2.3, Defined Constructed Data Types.

Important: This value should be placed in the Consumed Energy Odometer attribute (attribute 7) of the Base Energy Object that is pointed to by the Associated Energy Object Path attribute (attribute 41). This odometer shall never decrement.

#### 5B-7.3.1.2 Real Energy Generated Odometer, Attribute #2

The Real Energy Generated Odometer represents a range of 0 kWh to 999,999,999,999.999 kWh. The exponent base (n) shall be -3 represented as $10^{-3}$. See the defined constructed data type description of ODOMETER in Appendix C-6.2.3, Defined Constructed Data Types.

Important: This value should be placed in the Generated Energy Odometer attribute (attribute 8) of the Base Energy Object that is pointed to by the Associated Energy Object Path attribute (attribute 41). This odometer shall never decrement.

#### 5B-7.3.1.3 Real Energy Net Odometer, Attribute #3

The Real Energy Net Odometer represents a range of -999,999,999,999.999 kWh to 999,999,999,999.999 kWh. A positive value represents net real energy consumed. A negative value represents net real energy generated. The exponent base (n) shall be -3 represented as $10^{-3}$. See defined constructed data type description of SIGNED_ODOMETER in Appendix C-6.2.3, Defined Constructed Data Types.

#### 5B-7.3.1.4 Reactive Energy Consumed Odometer, Attribute #4

The Reactive Energy Consumed Odometer represents a range of 0 kVARh to 999,999,999,999.999 kVARh. The exponent base (n) shall be -3 represented as $10^{-3}$. The value represents reactive energy consumed (as in the case of an inductive load, load circuit with a lagging power factor, or generator with a leading power factor). See the defined constructed data type description of ODOMETER in Appendix C-6.2.3, Defined Constructed Data Types.

Important: This odometer shall never decrement.

#### 5B-7.3.1.5 Reactive Energy Generated Odometer, Attribute #5

The Reactive Energy Generated Odometer represents a range of 0 kVARh to 999,999,999,999.999 kVARh. The exponent base (n) shall be -3 represented as $10^{-3}$. The value represents reactive energy generated (as in the case of an capacitive load, load circuit with a leading power factor, or generator with a lagging power factor). See the defined constructed data type description of ODOMETER in Appendix C-6.2.3, Defined Constructed Data Types.

Important: This odometer shall never decrement.

#### 5B-7.3.1.6 Reactive Energy Net Odometer, Attribute #6

The Reactive Energy Net Odometer represents a range of -999,999,999,999.999 kVARh to 999,999,999,999.999 kVARh. A positive value represents net reactive energy consumed (as in the case of an inductive load, load circuit with a lagging power factor, or generator with a leading power factor). A negative value represents net energy generated (as in the case of a capacitive load, load circuit with a leading power factor, or generator with a lagging power factor). The exponent base (n) shall be -3 represented as $10^{-3}$. See defined constructed data type description of SIGNED_ODOMETER in Appendix C-6.2.3, Defined Constructed Data Types.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-7.3.1.7 Apparent Energy Odometer, Attribute #7

The Apparent Energy Odometer represents a range of 0 kVAh to 999,999,999,999.999 kVAh. The exponent base (n) shall be -3 represented as $10^{-3}$. See the defined constructed data type description of ODOMETER in Appendix C-6.2.3, Defined Constructed Data Types.

Important: This odometer shall never decrement

### 5B-7.3.1.8 Kiloampere-Hours Odometer, Attribute #8

The Kiloampere-Hours Odometer represents a range of 0 kAh to 999,999,999,999.999 kAh. The exponent base (n) shall be -3 represented as $10^{-3}$. See the defined constructed data type description of ODOMETER in Appendix C-6.2.3, Defined Constructed Data Type.

Important: This odometer shall never decrement.

### 5B-7.3.1.9 Total Real Power, Attribute #27

This attribute expresses the time rate of energy transfer as a REAL value, expressed in engineering units of kilowatts (kW). A positive value indicates consumption of energy, while a negative value indicates generation of energy.

Simple devices without the capability to measure energy are permitted to report in this attribute one or more nominal or user-defined power values that remain steady when the device is in an operating state and change infrequently relative to the rate at which the data is read from the instance. In this case the Real Energy Consumed Odometer (attribute #1) and the Real Energy Geerated Odometer (attribute #2) are not required.

It is encouraged that energy and energy transfer rate data are based on measured values.

### 5B-7.3.1.10 Associated Base Energy Object Path, Attribute #41

The Base Energy Object (class code 0x4E) acts as an “energy supervisor” for CIP Energy implementations. It is responsible for providing a time base for Energy integration, provides energy mode services, and provides aggregation services for aggregating energy values up through the enterprise. The Electrical Energy Object must be associated with a Base Energy Object. This attribute provides the mechanism for associating an instance of the Electrical Energy Object with an instance of the Base Energy Object. The class code and instance number of a Base Energy Object are encoded here. Consider a device that implements a single instance of the Base Energy Object and a single instance of the Electrical Energy Object. The value of this attribute would be:

03 00 21 00 4E 00 24 01

where: the first 2 bytes (03 00) indicate a padded EPATH of size 3 words, followed by the actual padded EPATH. See Appendix C of this specification for a complete discussion of EPATH encoding.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5B-7.1 Electrical Energy Object / Base Energy Object Association**

## 5B-7.4 Common Services

**Table 5B-7-4 Electrical Energy Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 01hex | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this objects attributes. |
| 03hex | Optional | Optional | Get_Attribute_List | Returns the contents of the selected gettable attributes. |
| 0Ehex | Required | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| See Appendix A for the definition of these services | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-7.4.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5B-7-5 Get_Attributes_All Response – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 [1] |
| 2 | UINT | Max Instance | 1 |
| 3 | UINT | Number of Instances | 1 |
| 6 | UINT | Max ID Number of Class Attributes | 0 |
| 7 | UINT | Max ID Number of Instance Attributes | 0 |

Table Footnotes:

1. The default value only applies for Revision 1 of this object. Newer revisions require that this attribute be implemented, which will override this default value.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attributes Values portion of the Get_Attributes_All response is as follows:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-7-6 Get_Attributes_All Response – Instance Level – Revision 01 and 02**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | ODOMETER | Real Energy Consumed Odometer | {0.0.0.0.0} |
| 2 | ODOMETER | Real Energy Generated Odometer | {0.0.0.0.0} |
| 3 | SIGNED_ODOMETER | Real Energy Net Odometer | {0.0.0.0.0} |
| 4 | ODOMETER | Reactive Energy Consumed Odometer | {0.0.0.0.0} |
| 5 | ODOMETER | Reactive Energy Generated Odometer | {0.0.0.0.0} |
| 6 | SIGNED_ODOMETER | Reactive Energy Net Odometer | {0.0.0.0.0} |
| 7 | ODOMETER | Apparent Energy Odometer | {0.0.0.0.0} |
| 8 | ODOMETER | Kiloampere-Hours Odometer | {0.0.0.0.0} |
| 9 | REAL | Line Freqency | 0.0 |
| 10 | REAL | L1-N Current | 0.0 |
| 11 | REAL | L2-N Current | 0.0 |
| 12 | REAL | L3-N Current | 0.0 |
| 13 | REAL | Average Current | 0.0 |
| 14 | REAL | Percent Current Unbalance | 0.0 |
| 15 | REAL | L1-N Voltage | 0.0 |
| 16 | REAL | L2-N Voltage | 0.0 |
| 17 | REAL | L3-N Voltage | 0.0 |
| 18 | REAL | Average L-N Voltage | 0.0 |
| 19 | REAL | L1-L2 Voltage | 0.0 |
| 20 | REAL | L2-L3 Voltage | 0.0 |
| 21 | REAL | L1-L3 Voltage | 0.0 |
| 22 | REAL | Average L-L Voltage | 0.0 |
| 23 | REAL | Percenet Voltage Unbalance | 0.0 |
| 24 | REAL | L1 Real Power | 0.0 |
| 25 | REAL | L2 Real Power | 0.0 |
| 26 | REAL | L3 Real Power | 0.0 |
| 27 | REAL | Total Real Power | 0.0 |
| 28 | REAL | L1 Reactive Power | 0.0 |
| 29 | REAL | L2 Reactive Power | 0.0 |
| 30 | REAL | L3 Reactive Power | 0.0 |
| 31 | REAL | Total Reactive Power | 0.0 |
| 32 | REAL | L1 Apparent Power | 0.0 |
| 33 | REAL | L2 Apparent Power | 0.0 |
| 34 | REAL | L3 Apparent Power | 0.0 |
| 35 | REAL | Total Apparent Power | 0.0 |
| 36 | REAL | L1 True Power Factor | 0.0 |
| 37 | REAL | L2 True Power Factor | 0.0 |
| 38 | REAL | L3 True Power Factor | 0.0 |
| 39 | REAL | Three Phase True Power Factor | 0.0 |
| 40 | UINT | Phase Rotation | 0 |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-7.5 Behavior

The behavior of static Electrical Energy Object instances is illustrated in Figure 5B-7.2 below. Note that if the Start_Metering and Stop_Metering services are not supported by the associated Base Energy Object instance (class code 0x4E), the Electrical Energy Object instance enters and remains in the Metering state. If the Start_Metering and Stop_Metering services are supported by the associated Base Energy Object instance (class code 0x4E), the Metering State is preserved through power cycles. The Metering State of an Electrical Energy Object instance always matches the Metering State of the associated Base Energy Object instance.

**Figure 5B-7.2 Static Electrical Energy Object State Diagram**

**Table 5B-7-7 Static Electrical Energy Object State Event Matrix**

| Event | State | |
|---|---|---|
| | Not Metering | Metering |
| Associated Base Energy Object Start_Metering Service sent | Transition to Metering | Ignore event |
| Associated Base Energy Object Stop_Metering service sent | Ignore Event | Transition to Not Metering |
| Associated Base Energy Object Type 0 Reset service sent | Reset all Odometers (attributes 1-8) values to {0.0.0.0.0}. | |
| Associated Base Energy Object Type 1 Reset service sent | Reset all Odometers (attributes 1-8) values to {0.0.0.0.0} and reset all other attribute values to default values. | |
| Associated Base Energy Object Type 2 Reset service sent | Return the Electrical Energy Object as closely as possible to the factory default configuration without resetting Energy Odometers (attributes 1 through 8). | |
| Get_Attributes_All | Validate and service the request and send the appropriate response | |
| Get_Attribute_List | | |
| Get_Attribute_Single | | |

The behavior of dynamic Electrical Energy Object instances is illustrated in Figure 5B-7.3 below. Note that if the Start_Metering and Stop_Metering services are not supported by the associated Base Energy Object instance (class code 0x4E), the Electrical Energy Object instance enters and remains in the Metering state upon creation. If the Start_Metering and Stop_Metering services are supported by the associated Base Energy Object instance (class code 0x4E), the Metering State is preserved through power cycles. The Metering State of an Electrical Energy Object instance always matches the Metering State of the associated Base Energy Object instance.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5B-7.3 Dynamic Electrical Energy Object State Diagram**

**Table 5B-7-8 Dynamic Electrical Energy Object State Event Matrix**

| Event | State | | |
|---|---|---|---|
| | **Non-Existent** | **Not Metering** | **Metering** |
| Create service to Associated Base Energy Object | Class instantiates an Electrical Energy Object. Transition to Metering | Not applicable | Not applicable |
| Delete service to Aassociated Base Energy Object | Error: Object does not exist (General Error Code 0x16) | Release all associated resources<br>Transition to Non-Existent | |
| Start_Metering service to Associated Base Energy Object | Error: Object does not exist (General Error Code 0x16) | Transition to Metering | Ignore event |
| Stop_Metering service to Associated Base Energy Object | Error: Object does not exist (General Error Code 0x16) | Ignore Event | Transition to Not Metering |
| Type 0 Reset to Associated Base Energy Object | Error: Object does not exist (General Error Code 0x16) | Reset all Odometer (attributes 1-8) values to {0.0.0.0.0}. | |
| Type 1 Reset service to Associated Base Energy Object | Error: Object does not exist (General Error Code 0x16) | Reset all Odometer (attributes 1-8) values to {0.0.0.0.0} and reset all other attribute values to default values | |
| Type 2 Reset service to Associated Base Energy Object | Error: Object does not exist (General Error Code 0x16) | Return the Electrical Energy Object as closely as possible to the factory default configuration without resetting Energy Odometers (attributes 1 through 8). | |
| Get_Attributes_All | Error: Object does not exist (General Error Code 0x16) | Validate and service the request and send the appropriate response | |
| Get_Attribute_List | | | |
| Set_Attribute_List | | | |
| Get_Attribute_Single | | | |
| Set_Attribute_Single | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5B-8 Non-Electrical Energy Object

## Class Code: 50 Hex

This object is used to provide unified non-electrical energy reporting capability for CIP enabled devices and processes. Energy reporting is typically related to the measurement of a variety of metering results. This object provides for the consistent reporting of non-electrical energy data, including, without limitation, natural gas, fuel oil, steam, compressed air, hot water, chilled water, etc.

Instances of the Non-Electrical Energy Object must be associated with an instance of the Base Energy Object (class code 0x4E). Instance attribute 11, “Associated Base Energy Object Path” is used to define this association.

## 5B-8.1 Revision History

Since the initial release of this object class definition, changes have been made that require a revision update of this object class. The table below represents the revision history:

**Table 5B-8-1 Non-Electrical Energy Object Revision History**

| Revision | Reason for object definition change |
|---|---|
| 01 | Initial definition at first release of specification |
| 02 | • Changed data types of energy odometer attributes 1 through 3 to use the newly defined SIGNED_ODOMETER and ODOMETER data types.<br>• Redefined path size in instance attribute 11 to be in WORDs instead of Bytes.<br>• Changed the Need In Implementation properties of energy odometer attributes 1-3 and attributes 6 & 10. |

## 5B-8.2 Class Attributes

**Table 5B-8-2 Non-Electrical Energy Object Class Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Required | Get | NV | Revision | UINT | Revision of this object | The current value assigned to this attribute is two (2) |
| 2 | Conditional[1] | Get | NV | Max Instance | UINT | Maximum instance number of an object currently created in this class level of the device | The largest instance number of a created object at this class hierarchy level. |
| 3 | Conditional[1] | Get | NV | Number of Instances | UINT | Number of object instances currently created at this class level of the device. | The number of object instances at this class hierarchy level. |
| 4 thru 7 | These class attributes are described in Volume 1, Chapter 4. Attributes 4 through 7 are optional. | | | | | | |

Table Footnotes:
1 Attributes 2 and 3 are required if the Create and Delete services are supported, or if more than one instance of the Non-Electrical Energy Object is supported


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-8.3 Instance Attributes

**Table 5B-8-3 Non-Electrical Energy Object Instance Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Conditional [1] | Get | NV | Energy Consumed Odometer | ODOMETER | The total energy consumed. | See section 5B-8.3.1.1 |
| 2 | Conditional [2] | Get | NV | Energy Generated Odometer | ODOMETER | The total energy generated. | See section 5B-8.3.1.2 |
| 3 | Optional | Get | NV | Energy Net Odometer | SIGNED_ODOMETER | The running total of Energy Consumed minus Energy Generated | See section 5B-8.3.1.3 |
| 4 | Required | Get | NV | Resource Type | UINT | Enumeration of type descriptor for the energy resource | See section 5B-8.3.1.4 |
| 5 | Required | Get | NV | Engineering Unit | ENGUNIT | The engineering units used for the energy odometers | See section 5B-8.3.1.5 |
| 6 | Conditional [3] | Get | NV | Engineering Unit Description | STRINGI | Text description of Engineering Unit e.g. “MJ” | See section 5B-8.3.1.6 |
| 7 | Required | Get | NV | Normalization Constant Multiplier | UINT | Conversion constant to Kilowatt-hours - multiplier | See section 5B-8.3.1.7 |
| 8 | Required | Get | NV | Normalization Constant Divisor | UINT | Conversion constant to Kilowatt-hours – divisor | See section 5B-8.3.1.8 |
| 9 | Conditional [4] | Get | NV | Energy Transfer Rate | REAL | | See section 5B-8.3.1.9 |
| 10 | Conditional [5] | Get | NV | Energy Flow Rate Time Base | ENGUNIT | | See section 5B-8.3.1.10 |
| 11 | Required | Get | NV | Associated Base Energy Object Path | SRUCT of: | Path to Energy Type Specific Object instance | See section 5B-8.3.1.11 |
| | | | | Path Size | UINT | Size of path (in words) | |
| | | | | Path | Padded EPATH | See Appendix C for format of this field. | |

Table Footnotes

1. Attribute 1 is required when Attribute 2 in the Base Energy Object instance that is pointed to by the Associated Energy Object Path attribute (attribute 41) is a value other than 4, “Energy Value Fixed” and the instance reports consumed energy.
2. Attribute 2 is required when Attribute 2 in the Base Energy Object instance that is pointed to by the Associated Energy Object Path attribute (attribute 41) is a value other than 4, “Energy Value Fixed” and the instance reports generated energy.
3. Attribute 6 is required when the value of Attribute 5 is “Vendor Specific” (0x0800-0x0FFF).
4. Attribute 9 is required only when Attribute 2 in the Base Energy Object instance that is pointed to by the Associated Energy Object Path attribute (attribute 41) is set to a value of 4, “Energy Value Fixed.”
5. Attribute 10 is required if Attribute 9 is present in the instance.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-8.3.1 Semantics

Attributes 1-3 all have a data type of ODOMETER or SIGNED_ODOMETER. The exponent base (n) shall be 0 represented as $10^0$. See defined constructed data type description of ODOMETER and SIGNED_ODOMETER in Appendix C-6.2.3.

The units associated with each odometer attribute will be given in the individual attribute semantics sections that follow. All energy odometers implemented in an instance of the non-electrical object shall use the same energy units.

When the energy odometers described in the following sections are implemented, energy values are computed periodically on an incremental basis. During each time period, the incremental consumed energy is added to the Energy Consumed odometer, the incremental generated energy is added to the Energy Generated odometer, and the difference between the incremental consumed and generated energy is added to the Energy Net odometer.

Note that an energy odometer value at a single point in time is of little value. The difference between the odometer values at two or more points in time is of interest, as the difference represents the amount of energy consumed by a process over the interval of time described by the two points. The consumer of energy odometer data must be able to discern and correctly interpret data rollover events so as to derive correct interval energy values.

#### 5B-8.3.1.1 Energy Consumed Odometer, Attribute #1

The total real energy consumed in the specified Engineering Units since the last energy accumulation reset. This value is always positive. The odometer array members have the following units.

#### 5B-8.3.1.2 Energy Generated Odometer, Attribute #2

The total real energy generated in the specified Engineering Units since the last energy accumulation reset. This value is always positive. The odometer array members have the following units.

#### 5B-8.3.1.3 Energy Net Odometer, Attribute #3

The Energy Consumed minus the Energy Generated, incrementally calculste each measuring period. A positive value indicates net energy consumed. A negative value indicates net energy generated. This value is a signed value.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-8.3.1.4 Resource Type, Attribute #4

This object provides for the consistent reporting of non-electrical energy data, including, without limitation, natural gas, fuel oil, steam, compressed air, hot water, chilled water, etc.

Resource Type values are enumerated in the following table.

**Table 5B-8-4 Energy Resource Types**

| ID | Resource Name |
|---|---|
| 0x0000 – 0x07FF | Reserved |
| 0x0800 – 0x0FFF | Vendor Specific |
| 0x1000 | Not specified |
| 0x1001 | Electricity |
| 0x1002 | Natural Gas |
| 0x1003 | Compressed Air |
| 0x1004 | Steam, Saturated |
| 0x1005 | Steam, Superheated |
| 0x1006 | Chilled Water |
| 0x1007 | Hot Water |
| 0x1008 | Hot Hot Water |
| 0x1009 | Crude Oil |
| 0x100A | Fuel Oil #2 |
| 0x100B | Fuel Oil #5 |
| 0x100C | Fuel Oil #6 |
| 0x100D | Diesel Oil |
| 0x100E | Gasoline |
| 0x100F | Propane |
| 0x1010 | Biogas |
| 0x1011 | Coal, Anthracite |
| 0x1012 | Coal, Bituminous |
| 0x1013 | Coal, Sub-bituminous |
| 0x1014 | Coal, Lignite |
| 0x1015 | Tallow |
| 0x1016 – 0xFFFF | Reserved |

### 5B-8.3.1.5 Engineering Unit, Attribute #5

The engineering unit of the specified resource, e.g. Joules, MBtu, etc. Expressed as a ENGUNIT as described in Appendix D of this Specification.

### 5B-8.3.1.6 Engineering Unit Descriptor, Attribute #6

A text description of the engineering unit of the specified resource. Optional unless the value of Engineering Unit (Attribute ID 5) is 0x1000, “unspecified”.

### 5B-8.3.1.7 Normalization Constant Multiplier, Attribute #7

The multiplier value for the scaling formula to convert from Engineering Units to the normalized value expressed in Kilowatt-hours (kWh).

### 5B-8.3.1.8 Normalization Constant Divisor, Attribute #8

The divisor value for the scaling formula to convert from Engineering Units to the normalized value expressed in Kilowatt-hours (kWh).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The tables below list examples of multipliers and divisors for converting common energy engineering units to kilowatt-hours to 3 significant digit precision. The multiplier shall be applied before the divisor to maintain the optimal accuracy and precision. The following tables are derived from the GRI G3.1 Guidelines Environmental IP EN3[4] and specify conversions for energy sources to kilowatt-hours.

**Table 5B-8-5 Normalization Constants for Conversions to kWh**

| Coal | Normalized Constant Multiplier | Normalized Constant Divisor |
|---|---|---|
| Tonne (metric) | 7220 | 1 |
| Ton (short) | 6550 | 1 |
| Ton (long) | 7340 | 1 |

| Crude Oil | Normalized Constant Multiplier | Normalized Constant Divisor |
|---|---|---|
| Barrel | 1730 | 1 |
| Tonne (metric) | 12400 | 1 |
| Ton (short) | 11300 | 1 |
| Ton (long) | 12600 | 1 |

| Gasoline | Normalized Constant Multiplier | Normalized Constant Divisor |
|---|---|---|
| Gallon | 347 | 10 |
| Tonne (metric) | 12400 | 1 |

| Diesel | Normalized Constant Multiplier | Normalized Constant Divisor |
|---|---|---|
| Gallon | 383 | 10 |
| Tonne (metric) | 12000 | 1 |

| Fuel Oil | Normalized Constant Multiplier | Normalized Constant Divisor |
|---|---|---|
| Gallon | 40 | 1 |
| Tonne (metric) | 11200 | 1 |

| Natural Gas | Normalized Constant Multiplier | Normalized Constant Divisor |
|---|---|---|
| Therm | 293 | 10 |
| 1000 Cubic Feet (Mcf) | 307 | 1 |
| 1000 cubic M | 10800 | 1 |
| MMBtu | 293 | 1 |

4 G3.1 Guidelines are an update and completion of the third generation of GRI's Sustainability Reporting Guidelines. The Guidelines are the cornerstone of GRI's Reporting Framework which provides guidance on how organizations can disclose their sustainability performance. Guidelines can be downloaded at the following URL: http://www.globalreporting.org/ReportingFramework/G31Guidelines/


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5B-8.3.1.9 Energy Transfer Rate, Attribute #9

The Energy Transfer Rate is the time rate of energy consumption or generation. The number is expressed as a REAL value. The engineering unit of this value is expressed as unit per time period, where unit is the specified Engineering Unit and the time period is the Energy Flow Rate Time Base, Attribute ID 10.

#### 5B-8.3.1.10 Energy Transfer Rate Time Base, Attribute #10

The time component of the Energy Flow Rate engineering unit. Expressed as an ENGUNIT as described in Appendix D of this Specification.

#### 5B-8.3.1.11 Associated Base Energy Object Path, Attribute #11

The Base Energy Object (class code 0x4E) acts as an “energy supervisor” for CIP Energy implementations. It is responsible for providing a time base for energy integration, provides energy mode services, and provides aggregation services for aggregating energy values up through the enterprise. The Non-Electrical Energy Object must be associated with a Base Energy Object. This attribute provides the mechanism for associating an instance of the Non-Electrical Energy Object with an instance of the Base Energy Object. The class code and instance number of a Base Energy Object are encoded here. Consider a device that implements a single instance of the Base Energy Object and a single instance of the Non-Electrical Energy Object. The value of this attribute would be:

03 00 21 00 4E 00 24 01

where: the first 2 bytes (03 00) indicate a padded EPATH of size 3 words, followed by the actual padded EPATH. See Appendix C of this specification for a complete discussion of EPATH encoding.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5B-8.1 Non-Electrical Energy Object / Base Energy Object Association**

## 5B-8.4 Common Services

**Table 5B-8-6 Non-Electrical Energy Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 01hex | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this objects attributes. |
| 03hex | Optional | Optional | Get_Attribute_List | Returns the contents of the selected gettable attributes. |
| 0Ehex | Required | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| See Appendix A for the definition of these services | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-8.4.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5B-8-7 Get_Attributes_All Response – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 [1] |
| 2 | UINT | Max Instance | 1 |
| 3 | UINT | Number of Instances | 1 |
| 6 | UINT | Max ID Number of Class Attributes | 0 |
| 7 | UINT | Max ID Number of Instance Attributes | 0 |

Table Footnotes:

1. The default value only applies for Revision 1 of this object. Newer revisions require that this attribute be implemented, which will override this default value.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5B-8-8 Get_Attributes_All Response – Instance Level – Revision 01 and 02**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | ODOMETER | Energy Consumed Odometer | {0.0.0.0.0} |
| 2 | ODOMETER | Energy Generated Odometer | {0.0.0.0.0} |
| 3 | SIGNED_ODOMETER | Energy Net Odometer | {0.0.0.0.0} |
| 4 | UINT | Resource Type | |
| 5 | ENGUNIT | Engineering Unit | |
| 6 | STRINGI | Engineering Unit Description | Number = 0 |
| 7 | UINT | Normalization Constant Multiplier | |
| 8 | UINT | Normalization Constant Divisor | |
| 9 | REAL | Energy Transfer Rate | 0.0 |
| 10 | ENGUNIT | Energy Transfer Rate Time Base | 0x1100 (second) |

Important: A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

### 5B-8.5 Behavior

The behavior of static Non-Electrical Energy Object instances is illustrated in Table 5B-8.9 below. Note that if the Start_Metering and Stop_Metering services are not supported by the associated Base Energy Object instance (class code 0x4E), the Non-Electrical Energy Object instance enters and remains in the Metering state. If the Start_Metering and Stop_Metering services are supported by the associated Base Energy Object instance (class code 0x4E), the Metering State is preserved through power cycles. The Metering State of a Non-Electrical Energy Object instance always matches the Metering State of the associated Base Energy Object instance.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5B-8.2 Static Non-Electrical Energy Object State Diagram**

**Table 5B-8-9 Non-Electrical Energy Object State Event Matrix**

| Event | State | |
|---|---|---|
| | **Not Metering** | **Metering** |
| Associated Base Energy Object Start_Metering Service sent | Transition to Metering | Ignore event |
| Associated Base Energy Object Stop_Metering service sent | Ignore Event | Transition to Not Metering |
| Associated Base Energy Object Type 0 Reset service sent | Reset all Odometer (attributes 1-3) values to zero | |
| Associated Base Energy Object Type 1 Reset service sent | Reset all Odometer (attributes 1-3) values to zero. and reset all other attribute values to default values. | |
| Associated Base Energy Object Type 2 Reset service sent | Reset the configuration attributes without resetting Odometers (attributes 1-3) | |
| Get_Attributes_All | Validate and service the request and send the appropriate response | |
| Get_Attribute_List | | |
| Get_Attribute_Single | | |

The behavior of dynamic Non-Electrical Energy Object instances is illustrated in Table 5B-8.10 below. Note that if the Start_Metering and Stop_Metering services are not supported by the associated Base Energy Object instance (class code 0x4E), the Non-Electrical Energy Object instance enters and remains in the Metering state upon creation. If the Start_Metering and Stop_Metering services are supported by the associated Base Energy Object instance (class code 0x4E), the Metering State is preserved through power cycles. The Metering State of an Non-Electrical Energy Object instance always matches the Metering State of the associated Base Energy Object instance.

**Figure 5B-8.3 Dynamic Non-Electrical / Base Energy Object State Diagram**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-8-10 Dynamic Non-Electrical Energy Object State Event Matrix**

| Event | State | | |
|---|---|---|---|
| | **Non-Existent** | **Not Metering** | **Metering** |
| Create service to Aassociated Base Energy Object | Class instantiates a Non-Electrical Energy Object. Transition to Metering | Not applicable | Not applicable |
| Delete service to Associated Base Energy Object | Error: Object does not exist (General Error Code 0x16) | Release all associated resources. Transition to Non-Existent | |
| Start_Metering service to Associated Base Energy Object | Error: Object does not exist (General Error Code 0x16) | Transition to Metering | Ignore event |
| Stop_Metering service to Associated Base Energy Object | Error: Object does not exist (General Error Code 0x16) | Ignore Event | Transition to Not Metering |
| Type 0 Reset service to Associated Base Energy Object | Error: Object does not exist (General Error Code 0x16) | Reset all Odometer (attributes 1-3) values to zero | |
| Type 1 Reset service to Associated Base Energy Object | Error: Object does not exist (General Error Code 0x16) | Reset all Odometer (attributes 1-3) values to zero. and reset all other attribute values to default values. | |
| Type 2 Reset service to Associated Base Energy Object | Error: Object does not exist (General Error Code 0x16) | Reset the configuration attributes without resetting Odometers (attributes 1-3) | |
| Get_Attributes_All | Error: Object does not exist (General Error Code 0x16) | Validate and service the request and send the appropriate response | |
| Get_Attribute_List | | | |
| Set_Attribute_List | | | |
| Get_Attribute_Single | | | |
| Set_Attribute_Single | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5B-9 Power Management Object

## Class Code: 53 Hex

Manufacturing equipment may sit idle for significant periods of time, waiting for a production request. Such periods may be over weekends, during idle shifts, at break times and even between scheduled operating cycles. Reducing the standby power can yield significant energy savings. A practical activity of facility energy management is to save energy by directing idle devices, machines or processes to go into a low power consumption state during known idle periods. Controllers or software tools can manipulate such a transition. The request to enter an energy saving state is also known as a pause request.

Another practical activity of facility energy management saving more energy for extended periods of time is by directing devices to go into a sleep state where even the device's communications hardware is sent to its lowest possible energy consumption state. For devices that support a sleep state, a special network wake mechanism is needed to wake the device.

The Power Management object provides standardized attributes and services to support the control of devices into and out of paused or sleep states.

A device supporting the Power Management Object can transition between various energy-related states. A Power Management service and optional adaptation specific sleep mechanisms are used to control entry into and exit from energy saving states. Within the paused states, a device may have multiple internal energy saving modes, each with a different power consumption level.

There are six basic energy-related states for CIP Power Management capable devices:

1. Power Off
2. Not Owned – Device is operational but no client "owns" the Power_Management service of the device.
3. Owned – The device is operational and a client "owns" the Power_Management service of the device.
4. Paused – Energy saving state; CIP communication continues.
5. Sleeping – Energy saving state; CIP communication is suspended.
6. Resuming – Device is transitioning from Paused to Owned.

Within a particular power management state, a device's operational capabilities and power levels may remain in transition for a time until the agreed power usage level is attained.

Within this object definition, the term "server" refers to a device that supports Power Management object capabilities, and the term "client" refers to the resource that manages the behavior of the server.

Although a controller is typically the client, a computer or another device may also manage energy saving states.

Please refer to Power Management Behavior for a specification of the server's response to a Power_Management service request.

The following figure illustrates the states available to a device that supports the Power Management Object. This diagram and its associated State Event Matrix will also appear in the Behavior section.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5B-9.1 State Transition Diagram**

Figure Footnotes
1. Adaptation-specific Wake from Sleep mechanism (example: Ethernet Magic Packet™

## 5B-9.1 Revision History

Since the initial release of this object class definition, changes have been made that require a revision update of this object class. The table below represents the revision history:

**Table 5B-9-1 Power Management Object Revision History**

| Revision | Reason for object definition update |
|---|---|
| 01 | Initial definition at first release of specification |
| 02 | Added path size to attribute 3<br>Get_Attributes_All response changed – because of change in size of attribute 3<br>Clarified Set_Pass_Code and Clear_Pass_Code service responses |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-9.2 Class Attributes

**Table 5B-9-2 Power Management Object Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Required | Get | NV | Revision | UINT | Revision of this object | The current value assigned to this attribute is two (02) |
| 2 thru 7 | These class attributes are optional and are described in Volume 1, Chapter 4 of the CIP Networks Library | | | | | | |

## 5B-9.3 Instance Attributes

**Table 5B-9-3 Power Management Object Instance Attributes**

| Attribute ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Required | Get[1] | V | Power Management Command | DWORD | Collection of bit fields comprising the most recent power management request | See Semantics section. |
| 2 | Required | Get | V | Power Management Status | DWORD | Collection of bit fields providing Power Management status information. | See Semantics section. |
| 3 | Required | Get | V | Client Path | STRUCT of: | Specifies the EPATH from this instance (server) to its current owner (client). | See Semantics section. |
| | | | | Path Size | UINT | Size of Path (in words) | |
| | | | | Path | Padded EPATH | See Appendix C for format of this field | |
| 4 | Required | Get | NV | Number of Power Management Modes | UINT | Number of Power Management Mode array entries in attribute 5 | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Power Management Object, Class Code: $53_{Hex}$

| Attribute ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 5 | Conditional [4] | Set[2]; Member Services[2] | NV | Power Management Modes | ARRAY of: | Array of Low Power Modes | See Semantics section. |
| | | | | Modes | STRUCT of: | Array of Mode Structures | |
| | | | | Minimum Pause Units | USINT | Specifies the Unit of Minimum Pause Time.<br>0 = millisecond<br>1 = second<br>2 = minute<br>3 = hour | |
| | | | | Minimum Pause Time | UINT | Minimum Accepted Pause Time | |
| | | | | Resume Units | USINT | Specifies the Unit of Resume Time.<br>0 = millisecond<br>1 = second<br>2 = minute<br>3 = hour | |
| | | | | Resume Time | UINT | Required Time to transition from the Paused State to the Owned state | |
| | | | | Power Level | REAL | Power in kW for this Mode | |
| | | | | Availability | BOOL | Specifies whether this Mode can be entered given current device state.<br>0 = unavailable<br>1 = available | |
| 6 | Required | Get | NV | Sleeping State Support | BOOL | The device supports the Sleeping state and the adaptation-specific Wake from Sleep mechanism<br>0 = not supported<br>1 = supported | See Semantics |
| 7 | Conditional[3] | Get | NV | Wake from Sleep Units | USINT | Specifies the Unit of Wake From Sleep Time.<br>0 = millisecond<br>1 = second<br>2 = minute<br>3 = hour | See Semantics |
| 8 | Conditional[3] | Get | NV | Wake from Sleep Time | UINT | Time to transition from the Sleeping state to the Paused state | See Semantics |

Table Footnotes:

1 The fields of the Power Management Command may only be modified by the object specific Power_Management service

2 Power Management Modes (Attribute #5) should only be settable on proxy devices using Set_Attribute_Single or Set_Member services (Refer to Base Energy Object Capability attribute = 2). For non-proxy devices, the Power Management Modes attribute should be a fixed read-only attribute (only Get_Single and Get_Member services should be supported).

3 These attributes are required if the value of attribute 6 (Sleeping State Support) is “Supported”.

4 Attribute #5 is required if the value of Attribute #4 is not equal to zero.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-9.3.1 Semantics

#### 5B-9.3.1.1 Power Management Command, Attribute #1

This attribute contains the most recent Power_Management service request from a client to a power management object instance in a server.

**Table 5B-9-4 Bit Field Definitions for Power Management Command**

| Bit | Name | Description | Data Type |
|---|---|---|---|
| 31-30 | Power Management Request | Specifies the nature of the most recent power management request to this instance<br>00 = Request status update only<br>01 = Request to Resume<br>10 = Request Pause<br>11 = Request Sleep | |
| 29 - 18 | Reserved | Reserved bits shall be set to 0 | |
| 17 - 16 | Unit of Requested Pause or Sleep Time | Specifies the unit of Requested Pause or Sleep Time.<br>00 = millisecond<br>01 = second<br>10 = minute<br>11 = hour | |
| 15 - 0 | Requested Pause or Sleep Time | Specifies the pause or sleep time requested of this instance using units specified by Unit of Requested Pause or Sleep Time | UINT |

The client shall modify fields in the Power Management Command attribute by using only the object specific Power_Management service (see section 5B-9.5.1) and read fields in the Power Management Status attribute to coordinate modification of the power management state of a server.

##### 5B-9.3.1.1.1 Power Management Request

The table below summarizes the coordination between the Power Management Request bit field and other fields contained in the Power Management Command.

**Table 5B-9-5 Coordination Among Fields in the Power Management Command**

| Power Management Request | Units of Requested Pause or Sleep Time | Requested Pause or Sleep Time | Notes |
|---|---|---|---|
| 00 | Required | Required | Update status request: May be used to preview the server's response to a pause or sleep request containing the specified parameter values. |
| 01 | Ignored | Ignored | Resume request |
| 10 | Required | Required | Request to transition to the Paused state. |
| 11 | Required | Required | Request to transition to the Sleeping state. |

##### 5B-9.3.1.1.2 Requested Pause or Sleep Time and Unit of Time

These fields together indicate the anticipated duration of a pause or sleep request.

A requested pause or sleep time value of zero shall indicate an indefinite time. The server will enter its lowest available energy saving mode within the Paused state, or the Sleeping state.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-9.3.1.2 Power Management Status, Attribute #2

This attribute contains energy related status information.

**Table 5B-9-6 Bits definitions for Power Management Status attribute**

| Bit | Name | Description |
|---|---|---|
| 31-29 | Present Power Management State | Specifies the current power management state of this instance.<br>000 = Not Owned<br>001 = Owned<br>010 = Paused<br>011 = Sleeping<br>100 = Resuming<br>101-111 = Reserved |
| 28 | Sleep Status | Specifies whether the instance will transition to Sleeping state in response to a Power_Management service. Bit is set only when Sleeping State Support (Attribute #6) has the value of Supported.<br>0 = Sleeping state not entered<br>1 = Sleeping state entered; client must use adaptation-specific Wake from Sleep method to wake the device. |
| 27 | | Reserved bit shall be set to 0 |
| 26-23 | Error Code | Specifies error status for the current power management request.<br>0000 = No error<br>0001 = Pass Code Not Set<br>0010 = Invalid Pass Code<br>0011 = Invalid request<br>0100 = Existing state same as requested state<br>0101 = Requested pause or sleep time is too short<br>0110 = State transition is prohibited by device state<br>0111 = State transition is prohibited by device configuration<br>1000 = Sleep requested but not supported<br>1001 - 1111 = reserved |
| 22 - 18 | Reserved | Reserved bits shall be set to 0 |
| 17 - 16 | Unit of Time to Resume | Specifies the unit of Time to Resume.<br>00 = millisecond<br>01 = second<br>10 = minute<br>11 = hour |
| 15 - 0 | Time to Resume | Specifies the time this instance requires to transition from Paused state to Owned state in response to a Request to Resume, in units specified by Unit of Time to Resume. The client must request the server to end its pause this amount of time before the end of the scheduled pause period. |

### 5B-9.3.1.3 Client Path, Attribute #3

The Client path returns the path to the current client that owns the server instance. The Set_Pass_Code service shall set the value of this attribute when a client establishes an ownership relationship with a server by setting a Pass Code. The Client Path may be used by other clients to identify the client that set the Pass Code. If no client currently owns the instance, the path shall be clear and the Path Size shall be zero.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5B-9.3.1.4 Power Management Modes, Attribute #5

The optional Power Management Modes attribute contains an array of structures describing the set of energy saving modes supported by the server. The array size is given in attribute #4 "Number of Power Management Modes". When a client issues a pause request, a device will enter the lowest energy mode that can be achieved within the requested pause time and is not in conflict with the device internal state. The low power modes of a device may be read for:

Display or informational purposes

Optimizing the selected pause time in order to minimize energy consumption

The Power Management Modes data structure contains an array element for each supported energy saving mode. The array elements are ordered with the lowest power level in the first element and increasing power levels in each subsequent element.

Each mode specifies the average Power Level that is achieved during the mode. The Power Level expresses the time rate of energy transfer as a REAL value, expressed in engineering units of kilowatts (kW). A positive value indicates consumption of energy, while a negative value indicates generation of energy.

The Pause Time describes the shortest requested pause that will allow entry into a mode. Pause Time is specified in units of Pause Units.

The Resume Time describes the maximum time required for a device to transition from the Paused state to the Owned state. Resume Time is specified in units of Resume Units.

The Availability member indicates whether a specific energy saving mode can be entered into, based on the device's internal state. A device may for example require outputs to be disabled prior to allowing entry to the lowest power mode, while allowing an output to be maintained during another low power mode.

#### 5B-9.3.1.5 Sleeping State Support, Attribute #6

This attribute indicates whether or not the server supports the Sleeping state.

#### 5B-9.3.1.6 Wake from Sleep Units and Time, Attributes #7 and #8

The Wake from Sleep Time describes the maximum time required for a device to transition from the Sleeping state to the Paused state. Wake from Sleep Time is specified in units of Wake from Sleep Units.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-9.4 Common Services

**Table 5B-9-7 Power Management Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 01hex | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this objects attributes. |
| 03hex | Optional | Optional | Get_Attribute_List | Returns the contents of the selected gettable attributes. |
| 04hex | n/a | Optional | Set_Attribute_List | Modifies the contents of the selected settable attributes. |
| 0Ehex | Conditional [1] | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 10hex | n/a | Conditional[2] | Set_Attribute_Single | Modifies an attribute value. |
| 18hex | n/a | Optional | Get_Member | Used to get individual power management modes in instance attribute 5 |
| 19hex | n/a | Optional | Set_Member | Used to set individual power management modes in instance attribute 5 |

Table Footnotes:
1 The Get_Attribute_Single service is Required if any Class attributes are implemented.
2 Required if any of the optional Set attributes are implemented.

See Appendix A for the definition of these services

### 5B-9.4.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5B-9-8 Get_Attributes_All Response – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 [1] |
| 2 | UINT | Max Instance | 1 |
| 3 | UINT | Number of Instances | 1 |
| 6 | UINT | Max ID Number of Class Attributes | 0 |
| 7 | UINT | Max ID Number of Instance Attributes | 0 |

Table Footnotes
1. The default value only applies for Revision 1 of this object. Newer revisions require that this attribute be implemented, which will override this default value.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-9-9 Get_Attributes_All Response – Instance Level – Revision 02**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | DWORD | Power Management Command | |
| 2 | DWORD | Power Management Satus | |
| 3 | STRUCT of UINT, Padded EPATH | Client Path | |
| 4 | UINT | Number of Power Management Modes | 0 |
| 5 | ARRAY of STRUCT | Power Management Modes | (null) |
| 6 | BOOL | Sleeping State Support | |
| 7 | USINT | Wake from Sleep Units | 0 |
| 8 | UINT | Wake from Sleep Time | 0 |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

**Table 5B-9-10 Get_Attributes_All Response – Instance Level – Revision 01 – (Obsolete)**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | DWORD | Power Management Command | |
| 2 | DWORD | Power Management Satus | |
| 3 | Padded EPATH | Client Path | |
| 4 | UINT | Number of Power Management Modes | 0 |
| 5 | ARRAY of STRUCT | Power Management Modes | (null) |
| 6 | BOOL | Sleeping State Support | |
| 7 | USINT | Wake from Sleep Units | 0 |
| 8 | UINT | Wake from Sleep Time | 0 |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

## 5B-9.5 Object Specific Services

**Table 5B-9-11 Energy Object Specific Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| $4D_{hex}$ | n/a | Required | Power_Management | Applies a power management request to a Power Management instance. |
| $4E_{hex}$ | n/a | Required | Set_Pass_Code | Used to set security pass code and establish energy management command ownership. This service is only accepted when a device is in the Not Paused and Not Owned state |
| $4F_{hex}$ | n/a | Required | Clear_Pass_Code | Used to clear security pass code and send a device to the Not Paused and Not Owned state. |

### 5B-9.5.1 Power_Management Service

This service applies a power management request to this Power Management Object instance. The service request data is made up of data contained in the Power Management Command attribute and a Pass Code. The service response data is the data contained in the Power Management Status attribute. This service provides a mechanism to carry out power management activities through explicit request/response messages.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

When a client is going to pause a server it generates a Power_Management service request. Detailed interaction transactions are depicted in the power management behavior section. Table 5B-9.12 defines the request parameters for the Power Management service.

**Table 5B-9-12 Power Management Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Power Management Command | DWORD | The power management command data in the format of instance attribute 1. | See section 5B-9.3.1.1, Power Management Command, Attribute #1 for the semantics of the Power Management Command attribute |
| Pass Code | DWORD | 32 bit value set by client to establish an ownership relationship with the device. 0 is reserved and treated as invalid. | |

Table 5B-9.13 defines the response parameters for the Power Management service.

**Table 5B-9-13 Power Management Response Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Power Management Status | DWORD | The power management status data in the format of instance attribute 2. | See section 5B-9.3.1.2, Power Management Status, Attribute #2 for the semantics of the Power Management Status attribute |

The Power_Management service request is Pass Code protected. The Pass Code is an integrity mechanism used to establish an exclusive relationship between a server and an authorized client or set of clients for power management. In order to modify a power state the client shall provide and maintain a Pass Code for the device.

The server shall not service explicit messages from other clients until the Power_Management service action is complete, the Power_Management service response has been sent, and the Power Management Status and other affected attributes have been updated.

### 5B-9.5.2 Set_Pass_Code Service

This service sets a Pass Code that establishes an ownership relationship between a client and a Power Management Object instance (server). When a server receives a Set_Pass_Code request, it shall not accept any other requests until the first request is processed. If the server is in the Not Owned state, e.g. its internally-stored Pass Code value is zero (0), and the Pass Code service parameter is not equal to zero (0), the server shall store the new Pass Code internally, enter the Owned state, and update the Power Management Status (Attribute #2) and Client Path (Attribute #3).

If the server is in the Owned state, e.g. its internally-stored Pass Code is not equal to zero (0), then:

1. If the Pass Code service parameter matches the internally-stored Pass Code and the Client Path service parameter matches the Client Path in Attribute #3, then an “already in requested mode/state” Error Response (0x0B) is returned.
2. Otherwise, an “object state conflict” Error Response (0x0C) is returned.

If the Pass Code service parameter is equal to zero, then an “invalid parameter” Error Response (0x20) is returned.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-9-14 Set_Pass_Code Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Pass Code | DWORD | 32 bit value set by client to establish an ownership relationship with the device. 0 is reserved and treated as invalid. | |
| Client Path | STRUCT of: | EPATH to the owning client. | See section 5B-9.3.1.3, Client Path, Attribute #3 for the semantics of the Client Path attribute |
| | UINT | Size of Path in words | |
| | Padded EPATH | See Appendix C for format of this field | |

The Client Path may be used in helping diagnose conflicts between power management clients. If it is required for more than one client to have access to the power management, it is up to the clients to share the Pass Code among them. The pass code is not exposed in the class or instance attributes.

If all clients that know the Pass Code become unavailable, the device can be recovered by cycling power. After power up, the server shall internally store a value of zero (0) for its Pass Code, set a null EPATH with Path Size of zero for its Client Path attribute, and enter the Not Owned state. Entry or exit from the Sleeping state or Paused state shall not clear the pass code or modify device ownership. If the power is cycled on the device unintentionally its status can be determined from the Power Management Status (instance attribute #2). If a device is found to be in the Not Owned state, then a client will need to re-establish the ownership relationship by setting the Pass Code again with the above procedure.

### 5B-9.5.3 Clear_Pass_Code Service

This service clears a server's Pass Code, thus terminating the ownership relationship between a client and a server. This service shall succeed only if a device is in the Owned state. When a server in the Owned state receives a Clear_Pass_Code request, if its internally-stored Pass Code value is equal to the Pass Code service parameter provided by the client, then the server shall internally store a value of zero (0) for its Pass Code, set a null EPATH with Path Size of zero for its Client Path (Attribute #3), enter the Not Owned state, and update the Power Management Status (Attribute #2).

If the Pass Code service parameter is equal to zero (0) or does not match the server's internally-stored Pass Code, then an "invalid parameter" Error Response (0x20) is returned.

If the server is in the Not Owned state, then an "already in requested mode/state" Error Response (0x0B) is returned.

If the server is in the Paused or Resuming state, then an "object state conflict" Error Response (0x0C) is returned.

**Table 5B-9-15 Clear_Pass_Code Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Pass Code | DWORD | 32 bit value equal to the established Pass Code value previously set, 0 is reserved and treated as invalid. | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-9.6 Behavior

A server supporting the Power Management Object can transition between various energy-related states. The Power_Management service and optional adaptation-specific sleep mechanisms are used to control entry into and exit from energy saving states. Within the paused state, a server may have multiple internal energy saving modes, each with a different power consumption level. A server may also have optional support for the Sleeping state to achieve a nearly complete shutdown.

### 5B-9.6.1 Power Management States

The following state transition diagram offers an illustration of the power management behavior of a server. Recall that the term "server" refers to a device that supports Power Management object capabilities, and the term "client" refers to the device or resource that manages the behavior of the device.

Explicit message common services such as Get_Attribute_Single, and the object-specific services Power_Management, Set_Pass_Code and Clear_Pass_Code, comprise the power management interface between a client and a server.

Refer to "Transactions between Client and Server" for more information on how explicit message services are used to control Power Management state transitions.

**Figure 5B-9.2 State Transition Diagram for Power Management Object**

Figure Footnotes

1. Adaptation-specific Wake from Sleep mechanism (example: Ethernet Magic Packet™)


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-9-16 Power Management Object State Event Matrix**

| Event | State | | | | | |
|---|---|---|---|---|---|---|
| | Power Off | Not Owned | Owned | Paused | Sleeping | Resuming |
| Power Up | Transition to Not Owned | Not Applicable | Not Applicable | Not Applicable | Not Applicable | Not Applicable |
| Power Down | Not Applicable | Transition to Power Off | Transition to Power Off | Transition to Power Off | Transition to Power Off | Transition to Power Off |
| Set Pass Code | Not Applicable | If Pass Code Service Parameter ≠ 0, Transition to Owned and update Pass Code Path and Power Management attributes; else Error: Invalid Parameter | If service parameters match those stored in server, then Error: Already in Requested State/Mode; else Error: Object State Conflict | Error: Object State Conflict | Ignore Event | Error: Object State Conflict |
| Clear Pass Code | Not Applicable | Error: Already in Requested State/Mode | If Pass Code is valid, Transition to Not Owned, clear the Pass Code and Path attributes; update the Power Management Status attribute; else Error: Invalid Parameter | Error: Object State Conflict | Ignore Event | Error: Object State Conflict |
| Update Status | Not Applicable | Update Power Management Status attribute | Update Power Management Status attribute | Update Power Management Status attribute | Ignore Event | Update Power Management Status attribute |
| Request Pause | Not Applicable | Error: Pass Code Not Set; Update Power Management Status attribute | If Request is valid[1], transition to Paused; else set applicable Error; update Power Management Status attribute | If Request is valid[1], remain in Paused and update requested pause time per new request; else set applicable Error; update Power Management Status attribute | Ignore Event | If Request is valid[1], transition to Paused; else set applicable Error; update Power Management Status attribute |
| Request Sleep | Not Applicable | Error: Pass Code Not Set; update Power Management Status attribute | If Request is valid[1], transition to Sleeping; else set applicable Error; update Power Management Status attribute | If Request is valid[1], transition to Sleeping; else set applicable Error; update Power Management Status attribute | Ignore Event | If Request is valid[1], transition to Sleeping; else set applicable Error; update Power Management Status attribute |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Event | State | | | | | |
|---|---|---|---|---|---|---|
| | **Power Off** | **Not Owned** | **Owned** | **Paused** | **Sleeping** | **Resuming** |
| Wake from Sleep[2] | Not Applicable | Ignore Event | Ignore Event | Ignore Event | Begin adaptation-specific wake from Sleep; transition to Paused | Ignore Event |
| Request to Resume | Not Applicable | Error: Pass Code Not Set; update Power Management Status attribute | Error: Existing state same as requested state; update Power Management Status attribute | If Pass Code is correct, transition to Resuming; update Power Management Status attribute, else Error: Invalid Pass Code. | Ignore Event | Error: Existing state same as requested state; update Power Management Status attribute |
| Complete Resuming (Internal to Device) | Not Applicable | Not Applicable | Not Applicable | Not Applicable | Not Applicable | Transition to Owned; update Power Management Status attribute |
| Identity Object Type 0 Reset | Not Applicable | Perform Type 0 Reset | Error: Device State Conflict (General Error Code 0x10) | Error: Device State Conflict (General Error Code 0x10) | Ignore Event | Error: Device State Conflict (General Error Code 0x10) |
| Identity Object Type 1 Reset | Not Applicable | Perform Type 1 Reset | Error: Device State Conflict (General Error Code 0x10) | Error: Device State Conflict (General Error Code 0x10) | Ignore Event | Error: Device State Conflict (General Error Code 0x10) |
| Identity Object Type 2 Reset | Not Applicable | Perform Type 2 Reset | Error: Device State Conflict (General Error Code 0x10) | Error: Device State Conflict (General Error Code 0x10) | Ignore Event | Error: Device State Conflict (General Error Code 0x10) |
| Get_Attributes_All | Not Applicable | Validate and service the request; send the appropriate response | | | Ignore Event | Validate and service the request and send the appropriate response |
| Get_Attribute_List | | | | | | |
| Set_Attribute_List | | | | | | |
| Get_Attribute_Single | | | | | | |
| Set_Attribute_Single | | | | | | |
| Get_Member | | | | | | |
| Set_Member | | | | | | |

1 A valid Pause or Sleep Request includes a correct Pass Code and Power Management Command parameters that do not cause the device to return an error.
2 Adaptation-specific Wake Request from Deep Sleep mechanism (Example: Ethernet Magic Packet)

### 5B-9.6.2 Ownership Relationship between Client and Server

A server shall power up in the Not Owned state and will accept a Set_Pass_Code service request from a client, thereby establishing an ownership relationship. Refer to the Object Specific Set_Pass_Code service definition for details.

With an ownership relationship established, the owner client or another client with knowledge of the Pass Code may utilize the Power_Management service to manage the energy states of the server.

An ownership relationship shall exist until a client with knowledge of the Pass Code uses the Clear_Pass_Code service to terminate the relationship, or power is cycled to the server.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-9.6.3 Transactions between Client and Server

Commanding a device to various power management states is done by a client. A client must "own" a device's Power_Management service before it can affect the energy-related states of a server. Once a client gains ownership of the Power_Management service, it can request energy-related state transitions as shown in the Energy State Transition Diagram in Figure 5B-9.2 and the State Event Matrix in Table 5B-9.16 Power Management Object State Event Matrix.

#### 5B-9.6.3.1 Time Elements in Power Management

The Power Management object includes a number of time elements.

- Requested Pause or Sleep Time – the time requested by a client issuing a Pause or Sleep; indicated the time until the server may be expected to have returned to a fully operational condition.
- Resume Time – the time required for a server to transition from the Paused state, through the Resuming state into the Owned state.
- Wake from Sleep Time – the time required for a server that supports the Sleeping state to transition from the Sleeping state to the Paused state.
- Minimum Pause or Sleep Time – the time required for a server to enter into and exit from an energy saving mode. This time includes the Wake from Sleep Time and Resume Time, as applicable, in addition to a time required to attain the agreed power usage level.

The figure below illustrates an example of time relationships associated with a successful Sleep request.

**Figure 5B-9.3 Example Time Relationship Diagram for Power Management Object**

When a server receives a Pause or Sleep request, it evaluates the Requested Pause or Sleep time against the total time needed to achieve a low power level, which may include an assessment of the server's available Modes and their associated Minimum Pause Times, plus the Wake from Sleep time, if applicable. If the Requested Pause or Sleep Time is shorter than this total time, the server shall deny the request and return a "Requested Pause or Sleep Time Too Short" error.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-9.6.3.2 Pause Request

The owner client may request a server to pause. To request a pause, the client shall pass a Power Management Request and the Pass Code as parameters to the Power_Management service. Refer to the Semantics of the Power_Management service for specifics.

A client may issue a Pause Request to a server in the Owned, Paused or Resuming states.

The server shall either accept or deny Power_Management service requests, and shall update the Error Code member of the Power Management Status service response parameter to indicate that the request is accepted or denied; and if denied shall indicate the reason. If the Error Code value is not equal to zero, the server state will not change. If the client request is accepted, the request is executed by the server, the Present Power Management State shall be set to (or remain in) ”Paused”, the Error Code shall be reset to 0, and Unit of Time to Resume and Time to Resume members shall be set appropriately. The server shall enter the lowest achievable energy saving mode consistent with the requested pause time as determined by the server’s algorithms.

The Time to Resume member of the Power_Management Status service response parameter indicates how long the server requires to reach the Owned state after a Request to Resume. The client needs to know this value in order to schedule a resume request, so that the device has sufficient time to transition to the Owned state in order that the actual pause duration does not exceed the desired pause duration.

In the event of a non-zero error code in the Power Management Status, the client may re-issue the request with modified parameters, or execute an application-specific error response routine.

### 5B-9.6.3.3 Sleep Request

The owner client may request a server to sleep. To do this, the client shall pass a Power Management request and the Pass Code as parameters to the Power_Management service. Refer to the Semantics of the Power_Management service for specifics.

Sleeping State Support (attribute #6) indicates whether the Sleeping state and associated wake mechanism are supported by the server.

A device in the Sleeping state consumes less power than a device in the Paused state, but an adaptation-specific “Wake from Sleep” network mechanism is needed to wake a device from the Sleeping state.

A client may issue a Sleep Request to a server in the Owned, Paused or Resuming states. A client that does not support the Wake from Sleep mechanism shall not issue a Sleep Request. The Sleeping state may be requested when the server and client both support the capability.

1. Sleeping State is entered when <u>all</u> of the following are true:
   a. The present state is Owned, Paused or Resuming
   b. The client supports Wake from Sleep commands
   c. The client issues a Sleep request
   d. The server supports the Sleeping state
   e. The server chooses to enter Sleeping state in response to a Power_Management request.
2. Sleeping state may not be entered when any of the following are true:
   a. The present state is Not Owned or Sleeping
   b. The client does not support Wake from Sleep commands


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

c. The server does not support Sleeping state

d. The client does not issue a Sleep request

e. The server chooses to not enter Sleeping state in response to a Sleep request.

The server shall either accept or deny Power_Management service requests, and shall update the Error Code member of the Power Management Status service response parameter to indicate that the request is accepted or denied; and if denied shall indicate the reason. If the Error Code value is not equal to zero, the server state will not change. If the client request is accepted, the request is executed by the server, the Present Power Management State shall be set to ”Sleeping”, the Error Code shall be reset to 0, and Unit of Time to Resume and Time to Resume members shall be set appropriately. The server shall enter the Sleeping state.

The Time to Wake attributes indicate how long the server requires to transition to the Paused state and recover its capacity to respond to CIP messaging. The Time to Resume member of the Power_Management Status service response parameter indicates how long the server requires to transition from the Paused state to the Owned state after a Request to Resume. The sum of the Time to Wake and Time to Resume indicates the minimum time the server requires to reach the Owned state from the Sleeping state. The client needs to know this value in order to schedule the Wake from Sleep.

In the event of a non-zero error code in the Power Management Status, the client may re-issue the request with modified parameters, or execute an application-specific error response routine.

Once the server enters the Sleep state, the client may close the communication connections to the server. The server shall not fault as a result of connections being closed when in a pause state. The client itself may then sleep or perform other application duties until it is needed to wake the server from the Sleeping state.

#### 5B-9.6.3.4 Wake from Sleep

Exit from the Sleeping state will require two distinct steps. First, the client shall employ an adaptation-specific Wake from Sleep mechanism (e.g. Magic Packet for an EtherNet/IP device) to transition the server to the Paused state, as CIP messages cannot be used to invoke the transition when there is no CIP communication stack running.

Following the Wake from Sleep step, a period of time may be required before a server is able to re-establish CIP messaging. This time is indicated by the Time to Wake Units and Time to Wake instance attributes. Once CIP communication is re-established with the server, the client can take the second step and issue a Request to Resume.

For more information about the Magic Packet for EtherNet/IP devices, refer to Volume 2, section 5-9, Power Management Object.

#### 5B-9.6.3.5 Request to Resume

At a point prior to the end of the pause the client sends a Request to Resume. The client shall use the Time to Resume and Unit of Time to Resume to calculate the resume time, subtracting this time from the desired time to end the pause.

If a server is in the Paused State, the client shall stop the pause using the Power_Management service. Power Management Command service parameter fields are passed along with the Pass Code. Refer to the Semantics of the Power_Management service for specifics.

A device in the “Paused” state transitions to the “Resuming” state when it has successfully responded to the Power_Management service Resume request. After the resume time has elapsed, the device will transition to the “Owned” state.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-9.6.3.6 Updating Power Management Status

A server shall update its Power Management Status attribute as indicated in the State Event Matrix.

## 5B-9.7 Power Management Messaging Sequences

Explicit messages are used to trigger power management activities. The illustrations that follow show several examples of messaging sequences.

**Figure 5B-9.4 Example: Client Requests a Pause**

The optional step "Check Validity of Request" uses the Request Status Update Only request of the Power_Management service to preview the server's anticipated response to a Pause Request. It is anticipated that upon receipt of the updated status, the client will immediately issue a pause request that takes into account the updated status of the server. The status of the server may change between the check validity step and the request pause step, in which case the server may deny the request and return an error.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5B-9.5 Example: Client Requests a Server to Resume**

**Figure 5B-9.6 Example: Client Requests a Pause Time That Is Too Short**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5B-9.7 Example: Client Issues Sleep Request**

**Figure 5B-9.8 Example: Client Issues Wake from Sleep Request**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-9.8 Modbus Integration Considerations

Support for the Power Management Object within the Modbus Translator will be defined in the future. There is ongoing work occurring in the Modbus Integration Special Interest Group.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5B-10 Power Curtailment Object

## Class Code: 5C Hex

The need exists in industrial applications for a standardized method of communicating with control devices to support the automated management of energy consumption and power levels. The Power Curtailment Object specifies a standardized device-level interface, based upon the CIP protocol, that is designed to meet this need.

In today's industrial environment, an application may utilize a number of methods to control the operation of devices, including direct analog and digital control, scheduled and unscheduled communications, etc. The Power Curtailment Object interface provides an alternate, standardized method that supplements customary control methods, for the purpose of controlling the energy usage and power level of industrial devices and systems. The interface is comprised of a discrete collection of power Curtailment Levels that can be selected by an energy management application.

**Definitions**

Managed Entity – A device that supports the Power Curtailment Object.

Energy Management Client ("Client") – A managing application that controls Managed Entities by interacting with instances of their Power Curtailment Objects.

Managed Instance – An instance of a Power Curtailment Object that may be owned by a Client. A Managed Entity can have multiple Managed Instances. If multiple Managed Instances are present, they should be independent of each other.

Power Curtailment State ("State") – A set of conditions that define behavior of a Managed Instance.

Curtailment Level – Collection of data representing a power level to which a Managed Instance can be commanded.

Pass Code – A client-created 32-bit unsigned integer that must be present in all Power Curtailment Object specific service requests. The Pass Code is a mechanism used to establish an integrity environment via an exclusive relationship between a server and an authorized client or set of clients for power curtailment.

**General**

The Power Curtailment Object can act as a proxy to provide the Power Curtailment Object interface for one or more simple devices that do not possess the capabilities to implement power Curtailment Levels.

A device that supports the Power Curtailment Object may also support other energy-related CIP objects.

A Energy Management Client can conserve energy by commanding one or more instances of the Power Curtailment Object to a Curtailment Level. Several options are available for providing versatility and for managing the content of a Power Curtailment Object instance's Curtailment Levels.

A Managed Instance can transition between various States and Curtailment Levels. The collection of object-specific services is used to control:

- Entry into and exit from Power Curtailment States
- Within the State, transition between Curtailment Levels


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

There are three Power Curtailment States for Managed Instances:

1. Nonexistent – The Managed Entity is powered down; or the Managed Instance has not yet been created.
2. Not Owned – The Managed Instance is operational but no Client has established an ownership relationship.
3. Managed – The Managed Instance is operational and a Client has established an ownership relationship.

A Managed Instance may have multiple defined Curtailment Levels. Every Managed Instance must contain at least one Curtailment Level that represents full power operation. The Client can respond to conditions, energy saving requests, etc., by selecting Curtailment Levels that exist in each member of its collection of Managed Instances, and by commanding each Managed Instance to transition to the selected Curtailment Level.

A Client can

- Configure Curtailment Levels in a Managed Instance.
- Manage transitions into and out of Power Curtailment States and Curtailment Levels
- Select an appropriate action for a given set of conditions

Within the Managed state, a Managed Instance's operational capabilities and power levels may remain in transition for a time until a commanded Curtailment Level is attained.

## 5B-10.1 Class Attributes

**Table 5B-10-1 Power Curtailment Object Class Attributes**

| Attribute ID | Need in Implementation | Access Rule | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|
| 1 | Optional | Get | Revision | UINT | Revision of this object | 01 |
| 2 | Conditional[1] | Get | Max Instance | UINT | Maximum instance number of an object currently created in this class level of the device. | The largest instance number of a created object at this class hierarchy level. |
| 3 | Conditional[1] | Get | Number of Instances | UINT | Number of object instances currently created at this class level of the device. | The number of object instances at this class hierarchy level. |
| 4 - 7 | These class attributes are optional and are described in Volume 1, Chapter 4 of the CIP Networks Library. | | | | | |

Table Footnotes:

1. Class Attributes 2 and 3 are required if the Create and Delete services are supported.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-10.2 Instance Attributes

**Table 5B-10-2 Power Curtailment Object Instance Attributes**

| Attribute ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Required | Get[1] | NV | Number of Curtailment Levels | UINT | Number of Curtailment Levels in attribute 2 | 1 - 10001<br>See section 5B-10.2.1.1 |
| 2 | Required | Get[1] | NV | Curtailment Levels | ARRAY of: | Array of Curtailment Levels | See section 5B-10.2.1.2 |
| | | | | Curtailment Level | STRUCT of: | | |
| | | | | ID | UINT | Unique number to identify the Curtailment Level | |
| | | | | Capabilities | WORD | Bit field defining the capabilities of the Curtailment Level. | |
| | | | | Status | WORD | Bit field defining the status of the Curtailment Level. | |
| | | | | Percent Power | UINT | Expected steady-state nominal power in % of Uncurtailed Power times 100 when operating at this Curtailment Level | |
| | | | | Description | STRINGI | Text Description of Curtailment Level | |
| | | | | Data Object Validity Check | UDINT | Vendor specific value that can be used to validate the integrity of the associated Data Object | |
| | | | | Data Object Path Size | UINT | Size of Path (in words) | |
| | | | | Data Object Path | Padded EPATH | See Appendix C for format of this field | |
| 3 | Required | Get | NV | Maximum Number of Curtailment Levels | UINT | Limit on number of curtailment levels | 1 – 10001<br>Default = 10001<br>See section 5B-10.2.1.3 |
| 4 | Required | Get | NV | Instance Capabilities | BYTE | Bit field Indicating Power Curtailment Capabilities | See section 5B-10.2.1.4 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Power Curtailment Object, Class Code: $5C_{Hex}$**

| Attribute ID | Need in Implementation | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 5 | Required | Get | V | Instance Status | STRUCT of | | See section 5B-10.2.1.5 |
| | | | | Matching ID | UINT | The ID of the Curtailment Level that best matches the results of a Go_To_Level service request | |
| | | | | Status Bits | WORD | Collection of bit fields providing status information | |
| 6 | Required | Set[2] | NV | Instance Options | BYTE | Bit field describing Options that control the behavior of a Managed Instance | See section 5B-10.2.1.6 |
| 7 | Required | Get | NV | Present Curtailment Level ID | UINT | ID of Curtailment Level at which the Managed Instance is currently operating | See section 5B-10.2.1.7 |
| 8 | Required | Get | NV | Present Expected Power | REAL | Present Power in kW | See section 5B-10.2.1.8 |
| 9 | Required | Get[1] | V | Owner Path | STRUCT of | Specifies the EPATH to this Managed Instance's owning client. | See section. 5B-10.2.1.9 |
| | | | | Path Size | UINT | Size of Path (in words) | |
| | | | | Path | Padded EPATH | See Appendix C for format of this field | |
| 10 | Required | Get | NV | Uncurtailed Power | REAL | Full Load Operating Power in kW | See section 5B-10.2.1.10 |

Table Footnotes:

1. Values of this attribute may be modified by object specific services.
2. Access shall be governed by the value of the Instance Options Settable bit in Instance Capabilities, Attribute #4

## 5B-10.2.1 Semantics

### 5B-10.2.1.1 Number of Curtailment Levels, Attribute #1

The Number of Curtailment Levels attribute specifies the number of members in the Curtailment Levels array of a Managed Instance.

### 5B-10.2.1.2 Curtailment Levels, Attribute #2

The Curtailment Levels attribute contains an array of structures describing the set of Curtailment Levels supported by the Managed Instance. The Curtailment Levels array consists of one member corresponding to each supported Curtailment Level. A Managed Instance shall contain, at minimum, one Curtailment Level (with ID = 0) for uncurtailed operation.

The Managed Instance shall keep the array of Curtailment Level sorted in descending order of the Percent Power member values.

A Curtailment Level may be specified in the Go_To_Level Service by use of its ID or by its Percent Power (percent of Uncurtailed Power times 100). The Curtailment Levels attribute members may also be read by the common Member services.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The Curtailment Levels of a device may be read for:

- Display or informational purposes
- Coordination of Curtailment Levels in multiple Managed Instances in order to control energy consumption

The ID member value shall be 0 or in the range of 30001 to 40000. Each Curtailment Level in the array shall have a unique ID value. Once a Curtailment Level is defined in a Managed Instance, its ID shall not be modifiable.

The Capabilities member indicates permissions for actions by a client on a defined Curtailment Level. The default value of 0x0000 gives an owner client full editing access to the Curtailment Level unless over-ridden by the value of Curtailment Level Array Read Only (Instance Capabilities, Attribute #4, bit 0).

**Table 5B-10-3 Bit Field Definitions for Capabilities Member**

| Bit | Name | Description |
|---|---|---|
| 0 | Hide Data Object Association | Governs whether the Curtailment Level Data Object Path member value in this Curtailment Level will be returned in a read of the Curtailment Level. 0 = not hidden (default), 1 = Hidden |
| 1 | Protected From Deletion | Governs whether this Curtailment Level may be deleted from the Managed Instance. 0 = Can be deleted (default), 1 = Cannot be deleted |
| 2 | Protected From Modification | Governs whether this Curtailment Level may be modified. 0 = Not protected (default), 1 = Protected. |
| 3 | Data Object Unnecessary | Indicates if a Data Object is needed for the instance to know how to transition to the Curtailment Level., 0 = Data Object needed, 1 = No Data Object needed (default value is vendor specific) |
| 4 – 15 | Reserved | Reserved bits shall be set to 0 |

The Status member provides status information on a defined Curtailment Level.

**Table 5B-10-4 Bit Field Definitions for Status Member**

| Bit | Name | Description |
|---|---|---|
| 0 – 3 | Data Status | 0 – Data Object Not Associated (default)<br>1 – Data Object Associated<br>2 – Capture Data Object in Progress<br>3 – Capture_Data Object Failed<br>4 to 15 – Reserved |
| 4 | Curtailment Level Unavailable | Indicates if Curtailment Level is available for transition. Set by Managed Entity.<br>0 = Level Available<br>1 = Level Not available. (default) |
| 5 | Curtailment Level Complete | Indicates if Curtailment Level is complete. Set by object specific services.<br>0 = Curtailment Level incomplete (default)<br>1 = Curtailment Level complete |
| 6 – 15 | Reserved | Reserved bits shall be set to 0 |

The Power member value has a permissible range of 1 to 10,000, reflecting 0.01 to 100.00 per cent of Attribute #10, Uncurtailed Power. For the Curtailment Level with ID = 0, this member value shall be 10000.

The Description member may be used to provide a multiple language text string that describes the Curtailment Level.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The Data Object Validity Check can be used by a Managed Instance to verify that a Data Object instance was not modified outside of the Power Curtailment Object environment. The content of this member shall be vendor specific. The Data Object Validity Check Supported bit in Instance Capabilities, Attribute #4 indicates support for the Data Object Validity Check by the Managed Instance.

- If the value of the Data Object Validity Check Supported bit = 0, then the value of the Data Object Validity Check shall be zero.
- If the value of the Data Object Validity Check Supported bit = 1, then the value of the Data Object Validity Check shall be vendor specific.

The Data Object Path member value is the EPATH to an associated Data Object instance if one exists. If no Data Object instance is associated or not yet associated, the Path Length shall be zero (0) and the EPATH shall be null. If the value of the Hide Data Object Association bit in the Capabilities member is equal to one, this member will return a Path Size value of zero (0) in response to an object-specific Read_Level service request or a common service request, e.g. Get_Attributes_All. The Data Object instance contains all the information required for a Managed Instance to transition to the Curtailment Level. The contents and format of this Data Object instance are vendor specific. The Data Object instance may be an instance of the File Object or a vendor-specific CIP object instance.

### 5B-10.2.1.3 Maximum Number of Curtailment Levels, Attribute #3

The Maximum Number of Curtailment Levels attribute specifies the number of possible members in the Curtailment Levels array of a Managed Instance. The maximum number of members that the Curtailment Levels array can contain is 10001. A Managed Entity can restrict this upper limit to a lower number due to memory limitations, Data Object capacity, etc. Range = 1 to 10001. Default = 10001.

### 5B-10.2.1.4 Instance Capabilities, Attribute #4

The Instance Capabilities attribute indicates the capabilities of the Managed Instance to support various power curtailment functions.

**Table 5B-10-5 Bit Field Definitions for Instance Capabilities**

| Bit | Name | Description |
|---|---|---|
| 0 | Curtailment Level Array Read Only [1] | 0 = Permits client to modify the Curtailment Levels array and its members using object specific services.<br>1 = Client may not modify the array or its members. |
| 1 | Instance Options Settable | Governs the Instance Options attribute Access Rule. 0 = Get (default), 1 = Set |
| 2 | Data Object Validity Check Supported | 0 = Managed Instance does not support Data Object Validity Check<br>1 = Managed Instance supports Data Object Validity Check |
| 3 – 7 | Reserved | Reserved bits shall be set to 0 |

Table Footnotes:

1. A value of 1 overrides the values of bits 1 and 2 in each Curtailment Level's Capabilities member in Attribute #2

### 5B-10.2.1.5 Instance Status, Attribute #5

The Instance Status attribute contains power curtailment related status information. A client can use this information to determine items such as; current state, current or future Curtailment Level, if transitioning between Curtailment Levels, etc.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The Matching ID member of this attribute contains the ID of the Curtailment Level that best matches the results of a Go_To_Level service request. This field represents the Curtailment Level to which the Managed Instance is transitioning (In Transition = 1) or the current Curtailment Level (In Transition = 0).

**Table 5B-10-6 Bit definitions for Status Bits member of Instance Status attribute**

| Bit | Name | Description |
|---|---|---|
| 0 – 3 | Present State | Specifies the current Power Curtailment State of this instance.<br>0 – Not Owned<br>1 – Managed<br>2 to 15 – Reserved |
| 4 | In Transition | Specifies if Managed Instance is transitioning between Curtailment Levels, 0=transition complete, 1=in transition |
| 5 | Management Unavailable | A 1 indicates that the Managed Instance is not currently capable of accepting Go_To_Level service requests |
| 6 | Curtailment Level Array Full | A 1 indicates that the Curtailment Level array is filled to capacity and the no more members may be added to the array. |
| 7 | Curtailment Level Mismatch | A 1 indicates that the current device configuration has been changed by means other than the Power Curtailment Object, and no longer matches the configuration of the present Curtailment Level. |
| 8 | Capture in Progress | A 1 indicates that a Capture_Level operation is in progress. |
| 9 – 15 | Reserved | Reserved bits shall be set to 0 |

### 5B-10.2.1.6 Instance Options, Attribute #6

The Instance Options attribute provides selection of operational behaviors of a Managed Instance. The default value for this attribute is 0x00.

**Table 5B-10-7 Bit Field Definitions for Options attribute**

| Bit | Name | Description |
|---|---|---|
| 0 | Power up Curtailment Level | 0 = Power up to last Curtailment Level (default)<br>1 = Power up to uncurtailed power (ID = 0) |
| 1 | Release Uncurtailed | 0 = Release Ownership allowed at any Curtailment Level (default)<br>1 = Release Ownership only allowed at Curtailment Level with ID = 0 |
| 2 – 7 | Reserved | Reserved bits shall be set to 0 |

The Power up Curtailment Level bit controls what a Managed Instance does upon power on.

A Managed Instance that had been in a Curtailment Level other than Uncurtailed Power (ID = 0) upon loss of power shall power up at the Curtailment Level determined in the following way.

- If the Power up Curtailment Level field of the Instance Options attribute (instance Attribute #6) = 0, then the Managed Instance will resume operation at the Curtailment Level in effect prior to the removal of power.
- If the Power up Curtailment Level field of the Instance Options attribute (instance Attribute #6) = 1, then the Managed Instance will resume operation at the Uncurtailed Power Curtailment Level (ID = 0).

### 5B-10.2.1.7 Present Curtailment Level ID, Attribute #7

The Present Curtailment Level ID attribute contains the ID of the Curtailment Level at which the Managed Instance is currently operating. This attribute value shall only change when the transition to a new Curtailment Level is complete.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

When the value of this attribute is zero (0), the Managed Instance is operating without curtailment (Uncurtailed Power).

#### 5B-10.2.1.8 Present Expected Power, Attribute #8

The Present Expected Power attribute contains a calculated value to indicate the Managed Instance's present expected power consumption, expressed in units of kilowatts (kW). Its value is determined as follows.

**Equation 5B-10.1 Present Expected Power Equation**

$$\frac{Present\ Curtailment\ Level\ Percent\ Power * Uncurtailed\ Power}{10{,}000}$$

When operating with no curtailment, this attribute shall be equal to the Uncurtailed Power attribute.

#### 5B-10.2.1.9 Owner Path, Attribute #9

The Owner Path attribute contains the path to the current Energy Management Client that owns the Managed Instance. The Establish_Ownership or Change_Ownership services shall set the value of this attribute when a Client establishes, reestablishes or modifies an ownership relationship with a Managed Instance (The Managed Instance shall set the Path Size, in words, upon successful completion of the service). The Owner Path may be used by other clients to identify the client that set the Pass Code. If no client currently owns the Managed Instance, the path shall be clear and the Path Size shall be zero.

#### 5B-10.2.1.10 Uncurtailed Power, Attribute #10

The Uncurtailed Power attribute indicates the nominal, average real power consumed by or under control of the Managed Instance (in kW) when it is running at full operation.

### 5B-10.3 Common Services

**Table 5B-10-8 Power Curtailment Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0x01 | Optional | Optional | Get_Attributes_All | Returns a predefined listing of this objects attributes. |
| 0x03 | Optional | Optional | Get_Attribute_List | Returns the contents of the selected gettable attributes. |
| 0x04 | n/a | Optional | Set_Attribute_List | Modifies the contents of the selected settable attributes. |
| 0x08 | Optional | n/a | Create [1] | Instantiates a new instance of the Power Curtailment Object. See section 5B-10.3.2. |
| 0x09 | n/a | Optional | Delete [1] | Deletes an instance of the Power Curtailment Object |
| 0x0E | Conditional [2] | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 0x10 | n/a | Conditional [3] | Set_Attribute_Single | Modifies an attribute value. |
| 0x18 | n/a | Optional | Get_Member | Used to get individual Curtailment Levels in instance attribute #1 |

Table Footnotes:

1. These services shall be supported together
2. The Get_Attribute_Single service is Required if any Class attributes are implemented.
3. Required if any of the optional Set attributes are implemented.

See Appendix A for the definition of these services


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-10.3.1 Get_Attributes_All Response

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5B-10-9 Get_Attributes_All Response – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 |
| 2 | UINT | Max Instance | 1 |
| 3 | UINT | Number of Instances | 1 |
| 6 | UINT | Max ID Number of Class Attributes | 0 |
| 7 | UINT | Max ID Number of Instance Attributes | 0 |

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5B-10-10 Get_Attributes_All Response – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Number of Curtailment Levels | |
| 3 | UINT | Maximum Number of Curtailment Levels | |
| 4 | BYTE | Instance Capabilities | |
| 6 | BYTE | Instance Options | |
| 5 | STRUCT of | Instance Status | |
| | UINT | Matching ID | |
| | WORD | Status Bits | |
| 7 | UINT | Present Curtailment Level ID | |
| 8 | REAL | Present Expected Power | |
| 9 | STRUCT of | Owner Path | |
| | UINT | Path Size | |
| | Padded EPATH | Path | |
| 10 | REAL | Uncurtailed Power | |

### 5B-10.3.2 Create Service Description

The Create service creates a new instance of a Power Curtailment Object in the Managed Entity. The Create service shall load the default values listed in Table 5B-10.13 into the newly created instance attributes. Created instances shall be non-volatile. Table 5B-10.11 defines the request parameters for the Create service.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-10-11 Create Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Uncurtailed Power | REAL | Full Load Operating Power in kW | See section 5B-10.2.1.10 |

Table 5B-10.12 defines the response parameters for the Create service.

**Table 5B-10-12 Create Response Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Instance_Number | UINT | Instance Number Allocated | 1-65535 |

Table 5B-10.13 defines the default instance attribute values for a Created instance.

**Table 5B-10-13 Default Created Instance Attribute Values**

| Attribute ID | Name | Default Value | Semantics of Values |
|---|---|---|---|
| 1 | Number of Curtailment Levels | 1 | Only the required Uncurtailed Curtailment Level |
| 2 | Curtailment Levels | | See section 5B-10.2.1.2 |
| | Uncurtailed Curtailment Level | | |
| | ID | 0 | Uncurtailed |
| | Capabilities | 0x0002 | Uncurtailed Curtailment Level may not be deleted |
| | Status | 0x0010 | Curtailment Level Not Available, No Data Object Associated, Curtailment Level Incomplete |
| | Percent Power | 10000 | represents 100.00 % |
| | Description | 0 | |
| | Data Object Validity Check | 0 | |
| | Data Object Path | | None associated |
| | Path Size | 0 | |
| | Path | “” | |
| 3 | Maximum Number of Curtailment Levels | 10001 | The Managed Entity may write a value less than 10001 to this attribute. |
| 4 | Instance Capabilities | N/A | Curtailment Levels Read Only bit shall be 0. The Managed Entity determines the remaining values. |
| 5 | Instance Status | | |
| | Matching ID | 0x0000 | |
| | Status Bits | 0x0020 | Not Owned, Management Unavailable |
| 6 | Instance Options | N/A | The Managed Entity determines these values. |
| 7 | Present Curtailment Level ID | 0 | Uncurtailed |
| 8 | Present Expected Power | Uncurtailed Power | Service request parameter value |
| 9 | Owner Path | | Null path, Size = 0 |
| | Path Size | 0 | |
| | Path | “” | |
| 10 | Uncurtailed Power | Uncurtailed Power | Service request parameter value |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-10.4 Object Specific Services

This section describes the object specific services associated with the Power Curtailment Object. With the exception of the Establish_Ownership service, all other object specific services are Pass Code protected. The Pass Code is a mechanism used to establish an integrity environment via an exclusive relationship between a Managed Instance and an authorized client or set of clients for power curtailment. In order to use these services, the client needs to provide and maintain a Pass Code for each Managed Instance.

The CIP specification does not mandate that one service request completes execution prior to starting the execution of another service request. If a Managed Entity supports this capability, it is imperative that certain portions of these object specific services be made atomic in order to avoid operation on Curtailment Levels that contain partial data.

**Table 5B-10-14 Energy Object Specific Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 0x4B | n/a | Required | Establish_Ownership | Used to set Pass Code and establish ownership. This service is only accepted when a Managed Instance is in the Not Owned state. |
| 0x4C | n/a | Required | Release_Ownership | Used to clear an existing Pass Code and send a Managed Instance to the Not Owned state. |
| 0x4D | n/a | Required | Change_Ownership | Used to change the existing Pass Code. The existing Pass Code must be known to perform this service. |
| 0x4E | n/a | Required | Go_To_Level | Commands a Managed Instance to a Curtailment Level. |
| 0x4F | n/a | Required | Go_To_Level_Query | Test the validity and previews the result of a Go_To_Level command for a Managed Instance; also used for verifying ownership. |
| 0x50 | n/a | Conditional [1] | Capture_Level | Used to create a new Curtailment Level based on the current Managed Instance operating characteristics |
| 0x51 | n/a | Conditional [1] | Remove_Level | Used to delete a Curtailment Level in a Managed Instance |
| 0x52 | n/a | Conditional [1] | Revise_Level | Used to modify a Curtailment Level in a Managed Instance |
| 0x53 | n/a | Required | Read_Level | Used to upload a Curtailment Level from a Managed Instance to a client |
| 0x54 | n/a | Conditional [1] | Write_Level | Used to download a Curtailment Level to a Managed Instance from a client |

Table Footnotes:

1. Required if value of Curtailment Level Array Read Only bit in Instance Capabilities Attribute #4 = 0 for any Managed Instance

### 5B-10.4.1 Establish_Ownership Service

The Establish_Ownership service changes the state of a Managed Instance to Managed and sets a Pass Code and sets the Owner Path attribute.

#### 5B-10.4.1.1 Establish_Ownership Request

Table 5B-10.15 defines the request parameters for the Establish_Ownership service.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-10-15 Establish_Ownership Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Pass Code | UDINT | 32 bit value set by client to establish an ownership relationship with the Managed Instance. | 0 is reserved and treated as invalid. |
| Owner Path | STRUCT of | EPATH to the owning client. | See section 5B-10.2.1.9 Owner Path, Attribute #9 for the semantics of this structure. |
| | UINT | Path Size | Size of EPATH in words |
| | Padded EPATH | Path | See Appendix C for semantics of this member. |

The Owner Path can only be set during the Establish_Ownership or Change_Ownership service requests. The Owner Path may be used in helping diagnose conflicts between Energy Management Clients. If it is required for more than one client to share the ownership relationship with a Managed Instance, it is up to the clients to share the Pass Code among them. The Pass Code is not exposed in the class or instance attributes.

### 5B-10.4.1.2 Establish_Ownership Response

None.

### 5B-10.4.1.3 Establish_Ownership Pseudo Code

The following pseudo code details requirements associated with the Establish_Ownership service:

```
‘ Check Request size
IF Length of Request < 6 bytes  THEN
   Return Error Response 0x13 (Not Enough Data)

‘ Now that we know the Path Size member is present…
IF Length of Request < 6 + 2 * Request.OwnerPath.PathSize bytes  THEN
   Return Error Response 0x13 (Not Enough Data)

IF Length of Request > 6 + 2 * Request.OwnerPath.PathSize bytes  THEN
   Return Error Response 0x15 (Too Much Data)

IF Request.PassCode == 0 THEN
   Return Error Response 0x20 (Invalid Parameter)

IF Request.OwnerPath is invalid THEN
   Return applicable Error Response from following:
       {0x04 (Path segment error),
       0x05 (Path destination unknown)[Optional]}

IF InstanceStatus.StatusBits:PresentState == NotOwned THEN
   Stored PassCode = Request.PassCode
   OwnerPath.Path = Request.OwnerPath
   OwnerPath.PathSize = Size of Request.OwnerPath in words
   InstanceStatus.StatusBits:PresentState = Managed
   Return Success

‘ Present State is Managed
IF Stored PassCode = Request.PassCode THEN
   Return Error Response 0x0B (Already in requested mode/state)
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
Return Error Response 0x0C (Object state conflict)
```

### 5B-10.4.2 Release_Ownership Service

The Release_Ownership service changes the state of the Managed Instance to Not_Owned and clears its Pass Code and Owner Path attribute, thus terminating the ownership relationship with all clients. If the Release Uncurtailed field in Instance Options (Attribute #6) equals 1, the Managed Instance must be operating at the uncurtailed Curtailment Level for this service to succeed.

#### 5B-10.4.2.1 Release_Ownership Request

Table 5B-10.16 defines the request parameters for the Establish_Ownership service.

**Table 5B-10-16 Release_Ownership Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Pass Code | UDINT | 32 bit value equal to the established Pass Code value previously set. | 0 is reserved and treated as invalid. |

Since the Pass Code is required to execute the Release_Ownership service for a Managed Instance, if a Pass Code is lost in the client(s), the Managed Entity will need to be power cycled to clear the Pass Codes for Managed Instances. Upon power up, all Managed Instances in the Managed Entity shall enter the Not Owned state. After a power cycle, the status of each Managed Instance can be determined from the Instance Status (instance attribute #5). A client can re-establish the ownership relationship by using the Establish_Ownership service.

#### 5B-10.4.2.2 Release_Ownership Response

None

#### 5B-10.4.2.3 Release_Ownership Pseudo Code

The following pseudo code details requirements associated with the Release_Ownership service:

```
IF InstanceStatus.StatusBits:PresentState != Managed THEN
   Return Error Response 0x0B (Already in Requested Mode/State)

‘ Check Request size to make sure it is not too short
IF Length of Request < 4 bytes  THEN
   Return Error Response 0x13 (Not Enough Data)
IF Request.PassCode != Stored PassCode THEN
   Return Error Response 0x0F (Privilege Violation)

‘ Check Request size to make sure it is not too long
IF Length of Request > 4 bytes  THEN
   Return Error Response 0x15 (Too Much Data)
IF InstanceOptions:ReleaseUncurtailed = 1 THEN
   IF PresentCurtailmentLevelID != 0 THEN
      Return Error Response 0x0C (Object State Conflict)
Stored PassCode = 0
OwnerPath.Path = “”
OwnerPath.PathSize = 0
InstanceStatus.StatusBits:PresentState = NotOwned
Return Success
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-10.4.3 Change_Ownership Service

The Change_Ownership service changes a Managed Instance's Pass Code and its Owner Path attribute. As an example of the use of this service, an owner client has shared an instance's Pass Code with other clients, but requires exclusive access for a time. The client can issue a Change_Ownership request with a different Pass Code to lock out other clients. After its need for exclusive access is complete, it may change the Pass Code back, restoring access to the other clients.

#### 5B-10.4.3.1 Change_Ownership Request

Table 5B-10.17 defines the request parameters for the Change_Ownership service.

**Table 5B-10-17 Change_Ownership Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Old Pass Code | UDINT | 32 bit value equal to the established Pass Code value previously set. | 0 is reserved and treated as invalid. |
| New Pass Code | UDINT | 32 bit value set by client to replace existing Pass Code in the Managed Instance. | 0 is reserved and treated as invalid. |
| New Owner Path | STRUCT of | EPATH to the owning client. | See section 5B-10.2.1.9 Owner Path, Attribute #9 for the semantics of this structure. |
| | UINT | Path Size | Size of EPATH in words |
| | Padded EPATH | Path | See Appendix C for semantics of this member. |

The Owner Path can only be set during the Establish_Ownership or Change_Ownership service requests. The Owner Path may be used in helping diagnose conflicts between Energy Management Clients. If it is required for more than one client to share the ownership relationship with a Managed Instance, it is up to the clients to share the Pass Code among them. The Pass Code is not exposed in the class or instance attributes.

#### 5B-10.4.3.2 Change_Ownership Response

None.

#### 5B-10.4.3.3 Change_Ownership Pseudo Code

The following pseudo code details requirements associated with the Change_Ownership service:

```
IF InstanceStatus.StatusBits:PresentState != Managed THEN
   Return Error Response 0x0C (Object state conflict)

‘ Check Request size
IF Length of Request < 10 bytes THEN
   Return Error Response 0x13 (Not Enough Data)
IF Request.OldPassCode != Stored PassCode THEN
   Return Error Response 0x0F (Privilege Violation)

‘ Now that we know the Path Size member is present…
IF Length of Request < 10 + 2 * Request.NewOwnerPath.PathSize bytes  THEN
   Return Error Response 0x13 (Not Enough Data)
IF Length of Request > 10 + 2 * Request.NewOwnerPath.PathSize bytes  THEN
   Return Error Response 0x15 (Too Much Data)
IF Request.NewOwnerPath is invalid THEN
   Return applicable Error Response from following:
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
            {0x04 (Path segment error),
            0x05 (Path destination unknown)[Optional]}
Stored PassCode = Request.NewPassCode
OwnerPath.Path = Request.NewOwnerPath
OwnerPath.PathSize = Size of Request.NewOwnerPath in words
Return Success
```

### 5B-10.4.4 Go_To_Level Service

The Go_To_Level service can be used by a client to command a Managed Instance to a Curtailment Level. The service request data is made up of a Pass Code, a Command and a Curtailment Level.

#### 5B-10.4.4.1 Go_To_Level Request

Table 5B-10.18 defines the request parameters for the Go_To_Level service. Table 5B-10.19 defines the fields of the Command parameter.

**Table 5B-10-18 Go_To_Level Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Pass Code | UDINT | 32 bit value that must match the value previously set by client. | 0 is reserved and treated as invalid. |
| Command | WORD | The power curtailment command | See Table 5B-10.19 |
| Curtailment Level | UINT | Specifies the commanded power level in units of % of Uncurtailed Power times 100, or the ID, depending upon the value of the Transition Criterion, of the Command parameter. | 1 – 10000 or<br>0, 30001 – 40000 |

**Table 5B-10-19 Bit Field Definitions for Command Request Parameter**

| Bit | Name | Description | Data Type |
|---|---|---|---|
| 0 – 2 | Transition Criterion | 0 = Transition to Specified Curtailment Level ID<br>1 = Transition to nearest available Curtailment Level equal to or below Commanded Power Level<br>2 = Transition to nearest available Curtailment Level equal to or above Commanded Power Level<br>3 – 7 = Reserved | 3 bit enumeration |
| 3 – 14 | Reserved | Reserved bits shall be set to 0 | |
| 15 | Check Ownership | Used by Go_To_Level_Query to verify client ownership. Ignored by Go_To_Level service. 1 = check ownership, 0 = execute query. | |

The Transition Criterion (bits 0 - 2) defines how the Managed Instance shall determine the new Curtailment Level target.

#### 5B-10.4.4.2 Go_To_Level Response

None.

#### 5B-10.4.4.3 Go_To_Level Pseudo Code

The following pseudo code details requirements associated with the Go_To_Level service:

```
IF InstanceStatus.StatusBits:PresentState != Managed THEN
   Return Error Response 0x0C (Object state conflict)

‘ Check Request size to make sure it is not too short
IF Length of Request < 8 bytes  THEN
   Return Error Response 0x13 (Not Enough Data)
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
IF Request.PassCode != Stored PassCode THEN
   Return Error Response 0x0F (Privilege Violation)

‘ Check Request size to make sure it is not too long
IF Length of Request > 8 bytes  THEN
   Return Error Response 0x15 (Too Much Data)
IF Request.Command:ReservedBits != 0 THEN
   Return Error Response 0x20 (Invalid Parameter)
IF InstanceStatus.StatusBits:ManagementUnavailable == 1
      OR  InstanceStatus.StatusBits:CaptureInProgress == 1
      OR InstanceStatus.StatusBits:InTransition == 1 THEN
   Return Error Response 0x0C (Object state conflict)

SELECT Request.Command:TransitionCriteria
 CASE 0: ‘Transition to Specified ID
   IF Request.CurtailmentLevel not in range (0, 30001-40000) THEN
      Return Error Response 0x20 (Invalid Parameter)
   Search Curtailment Levels array for
      Request.CurtailmentLevel == CurtailmentLevel.ID
   IF no match THEN
      Return Error Response 0x28 (Invalid Member ID)

   IF CurtailmentLevel.Status:NotAvailable == 1
        OR CurtailmentLevel.Status:LevelComplete == 0 THEN
     Return Error Response 0x0C (Object State Conflict)

 CASE 1: ‘ <= Percent Power
   IF Request.CurtailmentLevel not in range (1-10000) THEN
      Return Error Response 0x20 (Invalid Parameter)

   Search Curtailment Level array for first available Curtailment Level where
      CurtailmentLevel.PercentPower <= Request.CurtailmentLevel
      AND CurtailmentLevel.Status:NotAvailable == 0
      AND CurtailmentLevel.Status:LevelComplete == 1
   IF no match THEN
      Return Error Response 0x28 (Invalid Member ID)

 CASE 2: ‘ >= Percent Power
   IF Request.CurtailmentLevel not in range (1-10000) THEN
      Return Error Response 0x20 (Invalid Parameter)

   Search Curtailment Level array for first available Curtailment Level where
      CurtailmentLevel.PercentPower >= Request.CurtailmentLevel
      AND CurtailmentLevel.Status:NotAvailable == 0
      AND CurtailmentLevel.Status:LevelComplete == 1
   IF no match THEN
      Return Error Response 0x28 (Invalid Member ID)

 CASE ELSE:
   Return Error Response 0x20 (Invalid Parameter)

IF InstanceCapability.DataObjectValidityCheckSupported == 1 THEN
  IF perform Data Object Validity Check fails THEN
     Return Error Response 0x09 (Invalid attribute value)
InstanceStatus.StatusBits:InTransition = 1
InstanceStatus.MatchingID = found Curtailment Level
Start transition to new Curtailment Level
Return success
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Once the Managed Instance has completed its transition to a Curtailment Level, it needs to update some of its attributes. The following pseudo code describes actions to be taken upon completion of transitioning to a new Curtailment Level:

```
PresentCurtailmentLevelID = found Curtailment Level
PresentExpectedPower = UncurtailedPower *
    CurtailmentLevel.PercentPower / 10000
InstanceStatus.StatusBits:LevelMismatch = 0
InstanceStatus.StatusBits:InTransition = 0
```

### 5B-10.4.5 Go_To_Level_Query Service

The Go_To_Level_Query service can be used to test the validity of a power curtailment command without actually executing it. Unlike the Go_To_Level service, the Go_To_Level_Query service shall not return an exception response for an invalid Command, but rather return detailed Status data.

The service response data contains a Status and Target Curtailment Level ID parameter.

#### 5B-10.4.5.1 Go_To_Level_Query Request

The Go_To_Level_Query service request parameters are identical to the Go_To_Level service request parameters and are defined in Table 5B-10.18 and Table 5B-10.19. In addition, a bit in the Command parameter can be set to simply verify ownership.

#### 5B-10.4.5.2 Go_To_Level_Query Response

Table 5B-10.20 defines the response parameters for the Go_To_Level_Query service. Table 5B-10.21 defines the format of the Status parameter.

**Table 5B-10-20 Go_To_Level_Query Response Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Status | WORD | Bit field describing errors encountered simulating the command | See Table 5B-10.21 for the semantics of this parameter |
| Target Curtailment Level ID | UINT | ID of Curtailment Level identified based on Transition Criterion | 0, 30001 – 40000;<br>65535 ( 0xFFFF) indicates that the Curtailment Level request parameter could not be resolved into a valid Curtailment Level ID |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-10-21 Bit definitions for Status response parameter**

| Bit | Name | Description |
|---|---|---|
| 0 | Invalid Transition Criterion | Transition Criteria outside the valid range of 0 – 2. |
| 1 | Invalid Percent Power | The requested Curtailment Level is outside the valid range of a Percent Power (1 – 10000). |
| 2 | Invalid Curtailment Level ID | The requested Curtailment Level is outside the valid range of an ID (0, 30001 – 40000). |
| 3 | Nonexistent Curtailment Level | The requested ID does not exist in the Managed Instance. |
| 4 | Transition Prohibited by State | Instance management unavailable |
| 5 | Transition Prohibited by Configuration | Matching Curtailment Level is incomplete |
| 6 | In Transition | The Managed Instance is currently transitioning to a Curtailment Level. |
| 7 | Nonzero Reserved Bits | One or more reserved bits in the requested Command were not zero |
| 8 | Curtailment Level not Available | The target Curtailment Level is currently not available (Not Available bit in Status member of Curtailment Level is 1) |
| 9 | Capture In Progress | The Managed Instance is currently capturing a Curtailment Level and cannot transition to a new Curtailment Level at this time. |
| 10 | Data Object invalid | The associated data object failed a validity check. |
| 11 | Not Enough Data | Service request does not contain enough bytes of data |
| 12 | Too Much Data | Service request contains too many bytes of data |
| 13 – 15 | Reserved | Reserved bits shall be set to 0 |

### 5B-10.4.5.3 Go_To_Level_Query Pseudo Code

The following pseudo-code details requirements associated with the Go_To_Level_Query service:

```
IF InstanceStatus.StatusBits:PresentState != Managed THEN
   Return Error Response 0x0C (Object state conflict)

‘ Check Request size to make sure it at least contains a PassCode and Command
IF Length of Request < 6 bytes  THEN
   Return Error Response 0x13 (Not Enough Data)
IF Request.PassCode != Stored PassCode THEN
   Return Error Response 0x0F (Privilege Violation)

‘ Initialize response parameters
Response.TargetLevelID = 0xFFFF ‘In case a valid ID is not specified
Response.Status = 0 ‘Start with no error condition

‘ look for Check Ownership first.
IF Request.Command:CheckOwnership == 1 THEN
   Return Status, TargetLevelID

‘ Check Request size to make sure it is not too short
IF Length of Request < 8 bytes  THEN
   Response.Status:NotEnoughData = 1

 ‘ Check Request size to make sure it is not too long
IF Length of Request > 8 bytes  THEN
   Response.Status:TooMuchData = 1
IF Request.Command:ReservedBits != 0 THEN
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
   Response.Status:NonzeroReservedBits = 1
IF InstanceStatus.StatusBits:ManagementUnavailable == 1 THEN
   Response.Status:TransitionProhibitedByState = 1
IF InstanceStatus.StatusBits:CaptureInProgress == 1 THEN
   Response.Status:CaptureInProgress = 1
IF InstanceStatus.StatusBits:InTransition == 1 THEN
   Response.Status:InTransition = 1

SELECT Request.Command:TransitionCriteria
 CASE 0: ‘ Curtailment Level ID
   IF Request.CurtailmentLevel not in range (0, 30001-40000) THEN
      Response.Status:InvalidLevelID = 1
   ELSE
      Search Curtailment Level array for
        Request.CurtailmentLevel = CurtailmentLevel.ID
      IF no match THEN
        Response.Status:NonexistentLevel = 1
      ELSE
        Response.TargetLevelID = Request.CurtailmentLevel
        IF CurtailmentLevel.Status:NotAvailable THEN
          Response.Status:LevelNotAvailable = 1
        IF CurtailmentLevel.Status:LevelComplete == 0 THEN
          Response.Status:TransitionProhibitedByConfiguration = 1

 CASE 1: ‘ <= Percent Power
   IF Request.CurtailmentLevel not in range (1-10000) THEN
      Response.Status:InvalidPowerPercentage = 1
   ELSE
      Search Curtailment Level array for first available Curtailment Level
        Where CurtailmentLevel.PercentPower <= Request.CurtailmentLevel
        AND CurtailmentLevel.Status:NotAvailable == 0
        AND CurtailmentLevel.Status:LevelComplete == 1
      If matching Curtailment Levels exist but none are available
          or complete THEN
        Response.Status:LevelNotAvailable = 1
      ELSE IF no match THEN
        Response.Status:NonexistentLevel = 1
      ELSE
        Response.TargetLevelID = found CurtailmentLevel.ID
 CASE 2: ‘ >= Percent Power
   IF Request.CurtailmentLevel not in range (1-10000) THEN
      Response.Status:InvalidPowerPercentage = 1
   ELSE
      Search Curtailment Level array for first available Curtailment Level
        Where CurtailmentLevel.PercentPower >= Request.CurtailmentLevel
        AND CurtailmentLevel.Status:NotAvailable == 0
        AND CurtailmentLevel.Status:LevelComplete == 1
      If no matching Curtailment Levels are available and complete THEN
        Response.Status:LevelNotAvailable = 1
      ELSE
        Response.TargetLevelID = found CurtailmentLevel.ID
 CASE ELSE:
   Response.Status:InvalidTransitionCriterion = 1

IF Response.TargetLevelID != 0xFFFF
   AND InstanceCapability.DataObjectValidityCheckSupported == 1 THEN
   IF perform Data Object Validity Check fails THEN
      Response.Status:DataObjectInvalid = 1
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
Return Status, TargetLevelID
```

### 5B-10.4.6 Capture_Level Service

The Capture_Level service can be used to add a new Curtailment Level to a Managed Instance based on the current operating characteristics. The service request provides all Curtailment Level members except the Status, Data Object Path and Data Object Validity Check, which shall be provided by the Managed Instance. If the Managed Instance is currently in transition to a Curtailment Level, this service request shall return an error.

A Managed Entity shall support the Capture_Level service if the Curtailment Level Array Read Only bit of the Instance Capabilities attribute of any Managed Instance is 0.

#### 5B-10.4.6.1 Capture_Level Request

Table 5B-10.22 defines the request parameters for the Capture_Level service.

**Table 5B-10-22 Capture_Level Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Pass Code | UDINT | 32 bit value that must match the value previously set by client. | 0 is reserved and treated as invalid. |
| Percent Power | UINT | Power in % of Uncurtailed Power kW times 100 for this Curtailment Level | Range: 1 – 10000 |
| Description | STRINGI | Text Description of Curtailment Level | Text description of Curtailment Level. Can be in multiple languages. |

#### 5B-10.4.6.2 Capture_Level Response

Table 5B-10.23 defines the response parameters supported by the Capture_Level service.

**Table 5B-10-23 Capture_Level Response Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Curtailment Level ID | UINT | The ID of the created Curtailment Level | 30001-40000 |

#### 5B-10.4.6.3 Capture_Level Pseudo Code

The following pseudo-code details requirements associated with the Capture_Level service:

```
IF InstanceStatus.StatusBits:PresentState != Managed THEN
   Return Error Response 0x0C (Object state conflict)

‘ Check Request size to make sure it is not too short
‘ Note: This check is quite complex due to STRINGI data type
IF Length of Request too short  THEN
   Return Error Response 0x13 (Not Enough Data)
IF Request.PassCode != Stored PassCode THEN
   Return Error Response 0x0F (Privilege Violation)

‘ Check Request size to make sure it is not too long
‘ Note: This check is quite complex due to STRINGI data type
IF Length of Request is too long THEN
   Return Error Response 0x15 (Too Much Data)
IF Request parameters out of range THEN
   Return Error Response 0x20 (Invalid Parameter)
IF InstanceStatus.StatusBits:InTransition == 1 THEN
   Return Error Response 0x0C (Object state conflict)
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
IF InstanceStatus.StatusBits:CurtailmentLevelArrayFull == 1 OR
      InstanceCapabilities:CurtailmentLevelArrayReadOnly THEN
   Return Error Response 0x02 (Resource Unavailable)
IF InstanceStatus.StatusBits:CaptureInProgress == 1 Then
   Return Error Response 0x0C (Object state conflict)

‘ Prevent another capture operation until this one is complete
‘ This will also prevent transition to another Curtailment Level until capture is
complete.
InstanceStatus.StatusBits:CaptureInProgress = 1

Insert a new member in the Curtailment Levels Array
NumberOfCurtailmentLevels = NumberOfCurtailmentLevels + 1
Response.CurtailmentLevelID =new unique ID for new Curtailment Level
NewCrutailmentLevel.ID = Response.CurtailmentLevelID
NewCurtailmentLevel.PercentPower = Request.PercentPower
NewCurtailmentLevel.Description = Request.Description
NewCurtailmentLevel.Status:LevelUnavailable = 1
NewCurtailmentLevel.Status:LevelComplete = 0
NewCurtailmentLevel.DataStatus:State = CaptureDataObjectInProcess
NewCurtailmentLevel.DataObjectPath = “”
NewCurtailmentLevel.DataObjectPathSize = 0

‘ Set Managed Instance attributes to reflect this new Curtailment Level
InstanceStatus.MatchingID = Response.CurtailmentLevelID
IF this capture has filled the Curtailment levels array THEN
   InstanceStatus.StatusBits:CurtailmentLevelArrayFull = 1
PresentExpectedCurtailmentLevelPower = Request.PercentPower * UncurtailedPower
      / 10000
PresentCurtailmentLevelID = Response.CurtailmentLevelID

Begin capturing Curtailment Level data

Return Response.CurtailmentLevelID
```

Once the capturing of the data is complete, the Managed Instance needs to update various attributes in the Power Curtailment Object instance. The following pseudo-code details requirements associated with the completion of the Capture_Level service:

```
IF save Data Object successful THEN
   NewCurtailmentLevel.DataObjectPath = Saved Data Object path
   NewCurtailmentLevel.DataObjectPathSize=Saved Data Object Path size
         in WORDs
   NewCurtailmentLevel.Status:DataStatus = DataObjectAssociated
   IF InstanceCapabilities:DataObjectValidityCheckSupported == 1 THEN
      Calculate Data Object Validity Check
      NewCurtailmentLevel.DataObjectValidityCheck =Validity Check Result
   ELSE
      NewCurtailmentLevel.DataObjectValidityCheck = 0
   NewCurtailmentLevel.Status:LevelComplete = 1
   InstanceStatus.StatusBits:LevelMismatch = 0
ELSE
   NewCurtailmentLevel.Status:DataStatus = CaptureDataObjectFailed

InstanceStatus.StatusBits:CaptureInProgress = 0
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-10.4.7 Remove_Level Service

The Remove_Level service is used to delete a Curtailment Level from the Curtailment Levels Array in a Managed Instance. The Curtailment Level with ID = 0 or Curtailment Levels with the Curtailment Level Capabilities Protected from Deletion bit = 1 may not be deleted.

A Managed Entity shall support the Remove_Level service if the Curtailment Level Array Read Only bit of the Instance Capabilities attribute of any Managed Instance is 0.

#### 5B-10.4.7.1 Remove_Level Request

Table 5B-10.24 defines the request parameters for the Remove_Level service.

**Table 5B-10-24 Remove_Level Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Pass Code | UDINT | 32 bit value that must match the value previously set by client. | 0 is reserved and treated as invalid. |
| Curtailment Level ID | UINT | ID of Curtailment Level to be removed | 30001 – 40000 |

#### 5B-10.4.7.2 Remove_Level Response

None

#### 5B-10.4.7.3 Remove_Level Pseudo Code

The following pseudo-code details requirements associated with the Remove_Level service:

```
IF InstanceStatus.StatusBits:PresentState != Managed THEN
   Return Error Response 0x0C (Object state conflict)
‘ Check Request size to make sure it is not too short
IF Length of Request < 6 bytes  THEN
   Return Error Response 0x13 (Not Enough Data)
IF Request.PassCode != Stored PassCode THEN
   Return Error Response 0x0F (Privilege Violation)

‘ Check Request size to make sure it is not too long
IF Length of Request > 6 bytes  THEN
   Return Error Response 0x15 (Too Much Data)
IF Request parameters out of range THEN
   Return Error Response 0x20 (Invalid Parameter)
IF InstanceCapabilities:CurtailmentLevelArrayReadOnly THEN
   Return 0x2D (Instance Not Deletable)

‘ Avoid possibility of deleting a Curtailment Level that is being captured.
IF InstanceStatus.StatusBits:CaptureInProgress == 1 Then
   Return Error Response 0x0C (Object state conflict)

IF InstanceStatus.StatusBits:InTransition == 1 Then
   Return Error Response 0x0C (Object state conflict)

‘ Don’t delete Curtailment Level if currently at that Curtailment Level.
IF Request.CurtailmentLevelID == PresentCurtailmentLevelID THEN
   Return Error Response 0x0C (Object state conflict)

Search Curtailment Level array for
      Request.CurtailmentLevelID == CurtailmentLevel.ID
IF no match THEN
   Return Error Response 0x28 (Invalid Member ID)
IF found CurtailmentLevel.Capabilities:ProtectedFromDeletion == 1 THEN
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
   Return 0x0F (Permission Denied)

‘ Everything’s OK, delete Curtailment Level and
‘ adjust number of Curtailment Levels.
Remove found Curtailment Level from Curtailment Levels array
NumberOfCurtailmentLevels = NumberOfCurtailmentLevels – 1
Return Success
```

### 5B-10.4.8 Revise_Level Service

The Revise_Level service allows a client to revise the members associated with an existing Curtailment Level. Three options are available for treatment of the Data Object Path member: 1 – no change, 2 – replace with new path, and 3 – replace with new captured data.

A Managed Entity shall support the Revise_Level service if the Curtailment Level Modifiable bit of the Instance Capabilities attribute of any Managed Instance is 1.

#### 5B-10.4.8.1 Revise_Level Request

Table 5B-10.25 defines the request parameters for the Revise_Level service.

**Table 5B-10-25 Revise_Level Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Pass Code | UDINT | 32 bit value that must match the value previously set by client. | 0 is reserved and treated as invalid. |
| Curtailment Level ID | UINT | ID of Curtailment Level to revise. | Range: 0, 30001 - 40000 |
| Data Object Rule | UINT | Enumeration that specifies how to update the Data Object Path | 0 = existing Data Object remains unchanged<br>1 = Replace Data Object Path with Path contained in this service request<br>2 = Capture a new Data Object for this Curtailment Level. |
| Add to Description | BOOL | Adds to Description if set, replaces if not set | 1 = Add strings to existing Description, replacing those of same language<br>0 = Replace Description with contents of service request |
| Percent Power | UINT | Power in % of Uncurtailed Power kW times 100 for this Curtailment Level | Range: 1 – 10000 |
| Description | STRINGI | Curtailment Level Text Description | |
| Data Object Path | STRUCT of | Conditional, only required if Data Object Rule parameter = 1. | Contains EPATH to new Data Object. |
| | UINT | Path Size | Size of EPATH in words |
| | Padded EPATH | Path | See Appendix C for semantics of this member. |

The Add to Description parameter is used to instruct the Managed Instance on how to process the Description parameter. If it is zero, the Description parameter is simply copied over the current Description member of the target Curtailment Level. If it is one, the Description parameter is merged with the current contents of the Description member in the target Curtailment Level. For example, if the current Curtailment Level Description member contains strings in English, German and French, and the Description parameter of the service request contains strings in English, Chinese and Italian, the English string in the service request would replace the English string in the current Curtailment Level, and then the Chinese and Italian strings would be added to the Description attribute. This would result in the target Curtailment Level’s Description member with five strings, English, German, French, Chinese and Italian.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-10.4.8.2 Revise_Level Response

None.

### 5B-10.4.8.3 Revise_Level Pseudo Code

The following pseudo-code details requirements associated with the Revise_Level service:

```
IF InstanceStatus.StatusBits:PresentState != Managed THEN
   Return Error Response 0x0C (Object state conflict)
‘ Check Request size to make sure it is not too short
‘ Note: This check is quite complex due to STRINGI data type
‘ignore possibility of optional data path for now
IF Length of Request too short THEN
   Return Error Response 0x13 (Not Enough Data)
IF Request.PassCode != Stored PassCode THEN
   Return Error Response 0x0F (Privilege Violation)

‘ Check for Request tool long if no Data Object Path is present.
IF Request.DataObjectRule != 1 THEN
   ‘ Check Request size to make sure it is not too long
   ‘ Note: This check is quite complex due to STRINGI data type
   IF Length of Request too long THEN
      Return Error Response 0x15 (Too Much Data)
ELSE
   ‘ Data Object Path present
   ‘ Check first that Path Size member is present
   IF Length of Request too short THEN
      Return Error Response 0x13 (Not Enough Data)
   ‘ Now that we know the Path Size member is present, check length to make
   ‘sure it contains sufficient bytes in the Path member
   ‘ Note: This check is quite complex due to STRINGI data type
   IF Length of Request too short THEN
      Return Error Response 0x13 (Not Enough Data)
   ‘ Now that we know the Path Size member is present, check length to make
   ‘sure it contains no extra bytes in the Path member
   ‘ Note: This check is quite complex due to STRINGI data type
   IF Length of Request too long THEN
      Return Error Response 0x15 (Too Much Data)

IF Request parameters out of range THEN
   Return Error Response 0x20 (Invalid Parameter)

‘ If needed, make sure the requested data object path is valid.
IF Request.DataObjectRule == 1 THEN
   IF Request.DataObjectPath is invalid THEN
      Return applicable Error Response from following:
               {0x05 (Path destination unknown),
               0x04 (Path segment error),
               0x16 (Object does not exist)}
‘ Prevent possibility of modifying a Curtailment Level
’ which is the transition destination
IF InstanceStatus.StatusBits:InTransition == 1 THEN
   Return Error Response 0x0C (Object state conflict)
IF InstanceCapabilities:CurtailmentLevelArrayReadOnly THEN
   Return Error Response 0x02 (Resource Unavailable)

‘ Prevent possibility of modifying a Curtailment Level being captured.
IF InstanceStatus.StatusBits:CaptureInProgress == 1 Then
   Return Error Response 0x0C (Object state conflict)
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
‘ Don’t modify Curtailment Level if currently at that Curtailment Level.
IF Request.CurtailmentLevelID == PresentCurtailmentLevelID THEN
   Return Error Response 0x0F (Permission Denied)

Search Curtailment Level array for
      Request.CurtailmentLevelID == CurtailmentLevel.ID
IF no match THEN
   Return Error Response 0x28 (Invalid Member ID)
IF found CurtailmentLevel.Capabilities:ProtectedFromModification == 1 THEN
   Return Error Response 0x0F (Permission Denied)

‘ If Data Object Path provided, make sure found Curtailment Level needs it
IF Request.DataObjectRule == 1 THEN
   IF Found CurtailmentLevel.Capabilities:DataObjectUnnecessary == 1 THEN
      Return Error Response 0x0C (Object State Conflict)

‘ Prevent use of this Curtailment Level during modification
Found CurtailmentLevel.Status:LevelComplete = 0
‘ Prevent a revise or capture operation until this one is complete
InstanceStatus.StatusBits:CaptureInProgress = 1

Found CurtailmentLevel.PercentPower = Request.PercentPower
IF Request.AddToDescription == 1 THEN
   Merge Request.Description with Found CurtailmentLevel.Description
ELSE
   Found CurtailmentLevel.Description = Request.Description
IF Request.DataObjectRule == 2 THEN
   Found CurtailmentLevel.DataStatus:State = CaptureDataObjectInProcess
   Found CurtailmentLevel.DataObjectPathSize = 0
   Found CurtailmentLevel.DataObjectPath = “”

   Initiate capturing Curtailment Level data
ELSE IF Request.DataObjectRule == 1 THEN
   Found CurtailmentLevel.DataObjectPath = Request.DataObjectPath
   Found CurtailmentLevel.DataObjectPathSize
         = size of Request.DataObjectPath in WORDs
   IF InstanceCapabilities:DataObjectValidityCheckSupported == 1 THEN
      Calculate Data Object Validity Check
      Found CurtailmentLevel.DataObjectValidityCheck =Validity Check Result
   ELSE
      Found CurtailmentLevel.DataObjectValidityCheck = 0
   Found CurtailmentLevel.Status:DataStatus = DataObjectAssociated

   InstanceStatus.StatusBits:CaptureInProgress = 0
   Found CurtailmentLevel.Status:LevelComplete = 1

Return Success
```

Once the capturing of the data is complete (if applicable) the Managed Instance needs to update various attributes in the Power Curtailment Object instance. The following pseudo-code details requirements associated with the completion of the Revise_Level service when capture of new data is requested:

```
IF save Data Object successful THEN
   Found CurtailmentLevel.DataObjectPath.Path = path to saved Data Object
   Found CurtailmentLevel.DataObjectPathSize
         = size of path to saved Data Object in WORDs
   IF InstanceCapabilities:DataObjectValidityCheckSupported == 1 THEN
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
      Calculate Data Object Validity Check
      Found CurtailmentLevel.DataObjectValidityCheck =Validity Check Result
   ELSE
      Found CurtailmentLevel.DataObjectValidityCheck = 0
   Found CurtailmentLevel.Status:DataStatus = DataObjectAssociated
   Found CurtailmentLevel.Status:LevelComplete = 1
ELSE
   Found CurtailmentLevel.Status:DataStatus = CaptureDataObjectFailed

InstanceStatus.StatusBits:CaptureInProgress = 0
```

### 5B-10.4.9 Read_Level Service

The Read_Level service is used to upload a single Curtailment Level structure from the stored list of Curtailment Levels to a client. The content of the associated Data Object instance pointed to by the Data Object Path member is not read using this service. The value of the Data Object Path member can be used by the client to read the Data Object using object, common or vendor specific services.

#### 5B-10.4.9.1 Read_Level Request

Table 5B-10.26 defines the request parameters for the Read_Level service.

**Table 5B-10-26 Read_Level Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Pass Code | UDINT | 32 bit value that must match the value previously set by client. | 0 is reserved and treated as invalid. |
| Curtailment Level ID | UINT | ID of Curtailment Level to be read | 0, 30001 – 40000 |

#### 5B-10.4.9.2 Read_Level Response

Table 5B-10.27 defines the response parameters supported by the Read_Level service.

**Table 5B-10-27 Read_Level Response Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| ID | UINT | ID of Curtailment Level being read | 0, 30001 – 40000 = Read specified Curtailment Level |
| Capabilities | WORD | Bit field defining the capabilities of the Curtailment Level Curtailment Level. | |
| Status | WORD | Bit field defining the status of the Curtailment Level Curtailment Level. | |
| Percent Power | UINT | Power in % of Uncurtailed Power kW times 100 for this level | |
| Description | STRINGI | Text Description of Curtailment Level Curtailment Level | |
| Data Object Path | STRUCT of | The Path to the Data Object associated with this Curtailment Level | |
| | Path Size | UINT | Size of EPATH in words; 0 if no Data Object associated or if Data Object association is hidden |
| | Path | Padded EPATH | See Appendix C for semantics of this member. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-10.4.9.3 Read_Level Pseudo Code

The following pseudo-code details requirements associated with the Read_Level service:

```
IF InstanceStatus.StatusBits:PresentState != Managed THEN
   Return Error Response 0x0C (Object state conflict)
‘ Check Request size to make sure it is not too short
IF Length of Request < 6 bytes  THEN
   Return Error Response 0x13 (Not Enough Data)
IF Request.PassCode != Stored PassCode THEN
   Return Error Response 0x0F (Privilege Violation)

‘ Check Request size to make sure it is not too long
IF Length of Request > 6 bytes  THEN
   Return Error Response 0x15 (Too Much Data)
IF Request parameters out of range THEN
   Return Error Response 0x20 (Invalid Parameter)

‘ Prevent possibility of reading a level being captured.
IF InstanceStatus.StatusBits:CaptureInProgress == 1 Then
   Return Error Response 0x0C (Object state conflict)

Search Curtailment Level array for
      Request.CurtailmentLevelID == CurtailmentLevel.ID
IF no match THEN
   Return Error Response 0x28 (Invalid Member ID)
Response.ID = Found CurtailmentLevel.ID
Response.Capabilities = Found CurtailmentLevel.Capabilities
Response.Status = Found CurtailmentLevel.Status
Response.PercentPower = Found CurtailmentLevel.PercentPower
Response.Description = Found CurtailmentLevel.Description

‘ Check if Data Object data should be hidden.
IF Found CurtailmentLevel.Capabilities:HideDataObjectAssociation = 0 THEN
   Response.DataObjectPath = Found CurtailmentLevel.DataObjectPath
   Response.DataObjectPath.Size = Found CurtailmentLevel.DataObjectPathSize
ELSE
   Response.DataObjectPath.Size = 0
   Response.DataObjectPath = “”

Return Response
```

### 5B-10.4.10 Write_Level Service

The Write_Level service allows a client to download a single Curtailment Level to a Managed Instance. If the Curtailment Level requires an associated Data Object, it must exist in the Managed Instance prior to the Write_Level service. The Write_Level service cannot be used to overwrite a Curtailment Level in a Managed Instance with the same ID. If the operation is successful, the Curtailment Level is added to the array of Curtailment Levels and the Number of Curtailment Levels is updated. If not, the data is discarded.

A Managed Entity shall support the Write_Level service if the Curtailment Level Array Read Only bit of the Instance Capabilities attribute of any Managed Instance is 0.

#### 5B-10.4.10.1 Write_Level Request

Table 5B-10.28 defines the request parameters for the Write_Level service.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-10-28 Write_Level Request Parameters**

| Parameter Name | Data Type | Description of Parameter | Semantics of Value |
|---|---|---|---|
| Pass Code | UDINT | 32 bit value that must match the value previously set by client. | 0 is reserved and treated as invalid. |
| Curtailment Level ID | UINT | ID of Curtailment Level to write. | 30001 –- 40000 |
| Capabilities | UINT | Bit field defining the capabilities of the Curtailment Level. | |
| Percent Power | UINT | Power in % of Uncurtailed Power kW times 100 for this level | Range: 0 –- 10000 |
| Description | STRINGI | Text Description of Curtailment Level | |
| Data Object Path | STRUCT of | Conditional, only required if Data Unnecessary field in Capabilities parameter = 0. | EPATH to existing Data Object. |
| | UINT | Path Size | Size of EPATH in words |
| | Padded EPATH | Path | See Appendix C for semantics of this member. |

### 5B-10.4.10.2 Write_Level Response

None.

### 5B-10.4.10.3 Write_Level Pseudo Code

The following pseudo-code details requirements associated with the Write_Level service:

```
IF InstanceStatus.StatusBits:PresentState != Managed THEN
   Return Error Response 0x0C (Object state conflict)
‘ Check Request size to make sure it is not too short
‘ Note: This check is quite complex due to STRINGI data type
‘ignore possibility of optional data path for now
IF Length of Request too short THEN
   Return Error Response 0x13 (Not Enough Data)
IF Request.PassCode != Stored PassCode THEN
   Return Error Response 0x0F (Privilege Violation)

‘ Check for Request too long only if no Data Object Path is present.
IF Request.Capabilities:DataObjectUnnecessary == 1 THEN
   ‘ Check Request size to make sure it is not too long
   ‘ Note: This check is quite complex due to STRINGI data type
   IF Length of Request too long THEN
      Return Error Response 0x15 (Too Much Data)
ELSE
   ‘ Data Object Path present
   ‘ Check first that Path Size member is present
   IF Length of Request too short THEN
      Return Error Response 0x13 (Not Enough Data)
   ‘ Now that we know the Path Size member is present, check length to make
   ‘sure it contains sufficient bytes in the Path member
   ‘ Note: This check is quite complex due to STRINGI data type
   IF Length of Request too short THEN
      Return Error Response 0x13 (Not Enough Data)
   ‘ Now that we know the Path Size member is present, check length to make
   ‘sure it contains no extra bytes in the Path member
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
   ‘ Note: This check is quite complex due to STRINGI data type
   IF Length of Request too long THEN
      Return Error Response 0x15 (Too Much Data)

IF Request parameters out of range THEN
   Return Error Response 0x20 (Invalid Parameter)
IF InstanceStatus.StatusBits:CurtailmentLevelArrayFull == 1 OR
      InstanceCapabilities:CurtailmentLevelArrayReadOnly THEN
   Return Error Response 0x0F (Permission Denied)

Search Curtailment Level array for
      Request.CurtailmentLevelID == CurtailmentLevel.ID
IF match THEN
   Return Error Response 0x21 (Write-once value or medium already written)

IF Request.Capabilities:DataObjectUnnecessary == 0 THEN
   IF Request.DataObjectPath is invalid THEN
     Return applicable Error Response from following:
               {0x05 (Path destination unknown),
               0x04 (Path segment error),
               0x16 (Object does not exist)}

‘ Write Request data into new Curtailment Level

NewCurtailmentLevel.Capabilities = Request.Capabilities
NewCurtailmentLevel.PercentPower = Request.PercentPower
NewCurtailmentLevel.Description = Request.Description
NewCurtailmentLevel.Status:LevelUnavailable = 1
IF Request.Capabilities:DataObjectUnnecessary == 0 THEN
   NewCurtailmentLevel.DataObjectPath = Request.DataObjectPath
   NewCurtailmentLevel.DataObjectPathSize =
         size of Request.DataObjectPath in WORDs
   IF InstanceCapabilities:DataObjectValidityCheckSupported == 1 THEN
   Calculate Data Object Validity Check
      NewCurtailmentLevel.DataObjectValidityCheck =Validity Check Result
   ELSE
      NewCurtailmentLevel.DataObjectValidityCheck = 0
   NewCurtailmentLevel.Status:DataStatus = DataObjectAssociated

ELSE
   ‘ Data object is not required.
   NewCurtailmentLevel.Status:DataStatus = DataObjectNotAssociated
   NewCurtailmentLevel.DataObjectPathSize = 0
   NewCurtailmentLevel.DataObjectPath = ”“
   NewCurtailmentLevel.DataObjectValidityCheck = 0

NewCurtailmentLevel.Status:LevelComplete = 1

‘ Now that the Level is configured, add it to the list of levels.
Insert new Curtailment Level in Curtailment Levels array
NumberOfCurtailmentLevels = NumberOfCurtailmentLevels + 1

Return success
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-10.5 Behavior

A Managed Entity supporting instances of the Power Curtailment Object can transition between various Power Curtailment States and Curtailment Levels. The object specific Go_To_Level service is used to transition from one Curtailment Level to another.

### 5B-10.5.1 Power Curtailment States

The following state transition diagram offers an illustration of the behavior of a Managed Instance. The Establish_Ownership and Release_Ownership object specific services control movement between the Not Owned and Managed states. The remaining object specific services can only be used when a Managed Instance is in the Managed state. Common services can be used in either the Not Owned or the Managed state.

**Figure 5B-10.1 State Transition Diagram for Power Curtailment Object**

**Table 5B-10-29 Power Curtailment Object State Event Matrix**

| Event | Non-existent | Not Owned | Managed |
|---|---|---|---|
| Power Up | Transition to Not Owned | Not Applicable | Not Applicable |
| Power Down | Not Applicable | Transition to Non-existent | Transition to Non-existent |
| Create | Create new Instance; Transition to Not Owned | Not Applicable | Not Applicable |
| Delete | Return Error: Object Does Not Exist | On Success, Delete Instance Transition to Non-Existent | Return Error: Object State Conflict |
| Establish_Ownership | Not Applicable | On Success, Transition to Managed | Return Error: Already in requested mode/state |
| Release_Ownership | Not Applicable | Error: Already in requested mode/state | On Success, Transition to Not Owned and clear Client Path attribute; |
| Change_Ownership | Not Applicable | Error: Object state conflict | On Success, Change to New Pass Code and update Client Path |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Event | Non-existent | Not Owned | Managed |
|---|---|---|---|
| Go_To_Level; Go_To_Level_Query; Capture_Level; Remove_Level; Revise_Level; Read_Level; Write_Level; Associate_Level; Disassociate_Level | Not Applicable | Error: Object state conflict | Validate and service the request and send the appropriate response |
| Get_Attribute_Single; Set_Attribute_Single; Get_Member; Set_Member | Not Applicable | Validate and service the request and send the appropriate response | Validate and service the request and send the appropriate response |

### 5B-10.5.2 Ownership Relationship between Client and Managed Instance

A Managed Instance shall power up in the Not Owned state. A Managed Instance that had previously been in a Curtailment Level other than Uncurtailed Power (ID = 0) when power was removed shall power up at the Curtailment Level determined by the value of the Power Up Curtailment Level bit in the Instance Options attribute (#6). Refer to section 5B-10.2.1.6 for details.

Following power up, the Managed Instance can accept an Establish_Ownership service request from a client, thereby establishing an ownership relationship. Refer to the object specific Establish_Ownership service definition, section 5B-10.4.1 for details.

Regardless of the power-up Curtailment Level, a client needs to establish ownership of a Managed Instance (using the Establish_Ownership service) prior to utilizing any other object-specific service.

With an ownership relationship established, the owner client or another client with knowledge of the Pass Code may utilize object-specific services to control the Power Curtailment States and Curtailment levels of the Managed Instance.

The Managed Instance shall remain in the Managed state until a client with knowledge of the Pass Code uses the Release_Ownership service to terminate the relationship, power is cycled to the devices or a Type 0 Reset occurs.

### 5B-10.5.3 Curtailment Level assignment

The general approach embodied in the Power Curtailment Object is that power curtailment may be performed by a client by selecting pre-defined, discrete Curtailment Levels in individual members of a collection of Managed Instances in order to attain power reduction objectives at a system level. There are numerous options available to device vendors, machine OEMs, and system designers to assign Curtailment Levels to suit a particular application. Examples include:

- Curtailment Levels and their IDs may be pre- assigned by device vendors and documented in product literature
- Curtailment Levels and their IDs may be assigned by a machine builder during control system design
- Curtailment Levels may be saved in a Managed Instance during system commissioning (by use of object-specific Capture_Level, Remove_Level, Revise_Level, Read_Level, Write_Level, Associate_Level and Disassociate_Level Services)

The above list is not intended to exhaustive or to limit how a particular Energy Management Client utilizes the Power Curtailment Object.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-10.5.4 Transactions between Client and Managed Instance

A client that has established ownership of a Managed Instance can perform many power curtailment related activities as shown in Table 5B-10.29 Power Curtailment Object State Event Matrix.

### 5B-10.5.5 Updating Status Information

A Managed Instance reports instance status information as well as Curtailment level status information. This status information can be used by a Managed Entity or Clients to understand the current operating characteristics of a Managed Instance. The following sections elaborate the updating of the various status fields.

#### 5B-10.5.5.1 Updating Curtailment Level Status Member

There are three fields in the Status member of each Curtailment Level within a Managed Instance:

- Data Status – indicates the current status of an associated Data Object. This field is updated during the execution of object specific services and upon completion of a capture operation. The value is only relevant if the Data Object Unnecessary field in the Capability member is zero (0).
- Level Unavailable – Indicates if a Managed Instance can transition to this Curtailment Level. This field is set to one (1) during execution of the Create service and other object specific services that create new Curtailment Levels within a Managed Instance. After that, it is the responsibility of a Managed Entity to control the value of this field. This allows a Managed Entity to decide which Curtailment Levels should be available at any given time.
- Level Complete – Indicates if a Curtailment Level is usable or if items are missing. This field is updated during execution of the object specific services as Curtailment Levels are modified, written or captured.

#### 5B-10.5.5.2 Updating Instance Status

There are seven fields in the Instance Status attribute.

- Matching ID – Indicates target Curtailment Level identified by last successful Go_To_Level service execution. The Go_To_Level service sets this field to the ID of the destination Curtailment Level.
- Present State – Indicates the current Power Curtailment State of this instance. This field is set upon power up and the Create service to Not Owned and during execution of the object specific Ownership services.
- In Transition – Indicates if a Managed Instance is transitioning to a Curtailment Level. This field is set to one (1) during the execution of a Go_To_Level service request. It is set to zero (0) by the Managed Instance once the transition is complete.
- Management Unavailable – Indicates that the Managed Instance is not currently capable of accepting Go_To_Level service requests. This field is set to one (1) by the Managed Instance during execution of the Create service. It is then up to the Managed Entity to update the value of this field based upon its condition and ability to allow or disallow power curtailment activities.
- Curtailment Level Array Full – Indicates that the Curtailment Level array is filled to capacity and that no more members may be added. This field is updated during the execution of object specific services that add and remove elements from the Curtailment Levels Array attribute.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- Level Mismatch - Indicates that the current device configuration does not match the configuration of the present Curtailment level. This field is set to zero (0) by the Managed Instance upon completion of a transition to a Curtailment Level. This field is set to one (1) by the Managed Entity if any configuration data associated with the present Curtailment Level of a Managed Instance is changed outside of the Managed Instance.
- Capture in Progress – Indicates that a Capture_Level operation is in progress. This field is set to one (1) during object specific services that capture current operational characteristics to a Data Object and is set to zero (0) by the Managed Instance upon completion of a capture operation.

## 5B-10.6 Power Curtailment Messaging Sequences

The following sections illustrate several examples of how a client may use common and object specific services of the Power Curtailment Object to manage energy usage within a device. The examples are based on some common use cases, and use messaging sequences between a client and a Managed Instance for demonstration purposes. These diagrams are illustrative and do not necessarily imply required client behavior.

### 5B-10.6.1 Establish Ownership

Figure 5B-10.2 demonstrates successful ownership of a Managed Instance by a client. An optional step (Verify Ownership) demonstrates a method that can be used by a client to verify that it is the current owner of that Managed Instance.

**Figure 5B-10.2 Example: Client Establishes Ownership of a Managed Instance**

### 5B-10.6.2 Reestablish Ownership

Following a loss of power, a Managed Instance initializes into the Not Owned state. In this state, any object specific service request (other than Establish_Ownership) will fail. Furthermore, if a client shares a Pass Code and another client uses this shared Pass Code to either relinquish ownership or to change the Pass Code, any object specific service requests by the first client will also fail. Figure 5B-10.3 demonstrates how a client can use the Go_To_Level_Query service (with the Check Ownership bit set in the Command parameter) to determine if its stored Pass Code is still the owner of the Managed Instance. Depending upon the response, the client could use the Establish_Ownership service to re-establish ownership.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5B-10.3 Example: Client Re-establishes Ownership after Power Loss**

## 5B-10.6.3 Preview a Power Curtailment Command Response

A client can use the Go_To_Level_Query service request to determine the validity of parameters it may wish to use in a Go_To_Level service request. The status returned may indicate the Curtailment Level ID that corresponds to the requested Curtailment Level, or a conflict or error that would not permit the execution of a Go_To_Level command. The additional power of the Go_To_Level_Query service request is that for Command parameters that do not specify a target Curtailment Level, but rather one above or below a requested power level, it will return the ID of the Curtailment Level that best matches the criteria. The client can then use this information to determine actual energy usage upon execution of those parameters in a Go_To_Level command. Figure 5B-10.4 shows a client executing a Go_To_Level_Query command and its associated response.

**Figure 5B-10.4 Client Queries a Managed Instance**

## 5B-10.6.4 Command to Curtailment Level

Once a client decides it wants to command a Managed Instance to a new Curtailment Level, it needs to send a Go_To_Level service request with the appropriate parameters. The previous example demonstrated a method for a client to audit the contents of a Go_To_Level command parameter set. Upon receipt of a valid Go_To_Level command, a Managed Instance will begin the transition to the new Curtailment Level and return a successful response to the request. Typically, the transition will take some time to complete. The client can periodically poll the Managed Instance using the Get_Attribute_Single service request of the Instance Status attribute to verify when the transition into the commanded Curtailment Level is complete. Figure 5B-10.5 demonstrates this scenario.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5B-10.5 Client Commands a Managed Instance to a Curtailment Level**

## 5B-10.6.5 Obtain Exclusive Access to a Managed Instance

An owner that has shared its Pass Code with other clients may determine that modifying the instance may require exclusive ownership of the instance for a time. The Change_Ownership command provides this ability. Once the maintenance of the Managed Instance is complete, the owner may change the Pass Code and owner path back to their previous values and inform the other clients.

**Figure 5B-10.6 Client Establishes Exclusive Access for a Period of Time**

## 5B-10.6.6 Capture a New Curtailment Level

One method to create Curtailment Levels within a Managed Instance is to place a device in the desired operating condition and then use the Capture_Level service to save this condition into a Curtailment Level. Adjustment of the operating condition of a device containing a Managed Instance is done by other means than the Power Curtailment Object.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 5B-10.7 Create New Curtailment Level in a Managed Instance**

## 5B-10.6.7 Backup Curtailment Levels

Curtailment Levels not inherent to a Managed Instance will need to be restored in the unlikely event that a Managed Entity needs to be replaced. Figure 5B-10.8 shows a method that can be used to backup Curtailment Levels within a Managed Instance. A client need not be the owner of the Managed Instance to perform this operation. Common services can be used to read the size of the Curtailment Levels array and to get the contents of each member of the array. If the Data Object Path size member of a Curtailment Level is non-zero, object, common or vendor specific services can be used to retrieve the associated Data Object contents. The client may also want to check the Status member to make sure that the Curtailment Level being backed up is complete and not in the process of being revised or missing its associated Data Object instance (if required).

**Figure 5B-10.8 Back Up Curtailment Levels from a Managed Instance**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-10.6.8 Restore Curtailment Levels

There may come a time when a client will need to restore Curtailment Levels that were present in a Managed Instance (e.g. device replacement). Figure 5B-10.9 demonstrates one method that could be used to accomplish this task. Note, a client must own the Managed Instance to perform this activity. The Curtailment Level with ID = 0 cannot be restored using the Write_Level service, since it is always present in a Managed Instance. Instead, the Revise_Level service should be used for this special Curtailment Level.

**Figure 5B-10.9 Restore Curtailment Levels in a Managed Instance**

### 5B-10.6.9 Create a New Power Curtailment Object Instance

If the optional Create and Delete common services are implemented, a client may create a new instance of the Power Curtailment Object by using the Create common service. The newly created instance includes the required Curtailment Level 0, however, in many cases, this Curtailment Level may not include sufficient detail to be operational.

Figure 5B-10.10 demonstrates a method by which a client can complete the definition of Curtailment Level 0 by use of the Revise_Level object-specific service, adding a Data Object Path, Description, and other members.

**Figure 5B-10.10 Create a New Instance and Complete Curtailment Level 0**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 5B-11 Connection Configuration Object

**Class Code: F3 Hex**

This object defines an interface used to create, configure and control CIP connections in a device. This specification does not define or constrain the operation of the device's connection management state machine.

The Port object shall be supported if the Connection Configuration object is supported and the device supports more than one CIP Port or the device supports a single CIP Port and the Port Number is not 2. The Port object is required because a client of the Connection Configuration object needs to be able to learn the Port Numbers of all the ports that can route the class 0/1 forward open services.

## 5B-11.1 Class Attributes

**Table 5B-11-1 Connection Configuration Object Class Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Value |
|---|---|---|---|---|---|---|---|
| 1 | Required | Get | NV | Revision | UINT | Third revision, value = 3 | See Chapter 5B-11.1.1.1 |
| 2 | Required | Get | NV | Max Instance | UDINT | Maximum instance number | |
| 3 | Required | Get | NV | Num Instances | UDINT | Number of connections currently instantiated | |
| 4 | This class attribute is optional and is described in Chapter 4 | | | | | | |
| 5 | Conditional [5] | Get | NV | Optional service list | Struct of: | List of optional services utilized in an object class implementation | A list of service codes specifying the optional services implemented in the device for this class. |
| | | | | Number services | UINT | Number of services in the optional service list | The number of service codes in the list |
| | | | | Optional services | Array of UINT | List of optional service codes | The optional service codes. |
| 6 and 7 | These class attributes are optional and are described in Chapter 4 | | | | | | |
| 8 | Required | Get | NV | Format Number | UINT | | See Semantics section 5B-11.1.1.2 |
| 9 | Required | Set | NV | Edit Signature | UDINT | | See Semantics section 5B-11.1.1.3 |
| 10 thru 32 | Reserved | | | | | | |
| 33 | Optional | Set [1 2] | NV | Scanner Mode | BOOL | The originator to target packets for connections originated by this device shall reflect the state of this attribute. | 0 = idle mode<br>1 = run mode |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Value |
|---|---|---|---|---|---|---|---|
| 34 | Optional | Get | V | Scanner Capabilities [4] | WORD | Bit 0, set to Scanner Mode (attribute 33) supported 3<br>Bit 1, set to Scanner Mode (attribute 33) currently supported<br>Bit 2, Instances may currently be created in Run mode<br>Bit 3, Instances may currently be changed in Run mode<br>Bit 4, Instances may currently be deleted in Run mode<br>Bit 5, Instance may currently be created in Idle mode<br>Bit 6, Instances may currently be changed in Idle mode<br>Bit 7, Instances may currently be deleted in Idle mode<br>Bit 8, Open_Connection/Close_ Connection services supported<br>Bit 9, Stop_Connection service supported<br>Bit 10, Get_Member service to read image tables supported<br>Bits 11-15 – reserved and shall be zero | Bits 0 – 10<br>= 0, No<br>= 1, Yes |

Table Footnotes:

1 A set to attribute 33 shall return a device state conflict (0x10) error if changing the Scanner Mode is not permitted when the set attribute command is received.

2 The value of attribute 33 may be different than the value last set to attribute 33 if another mechanism (like a key switch) changed the scanner's mode.

3 Bit 0 indicates if the Scanner Mode attribute has the ability to ever be set using a set attribute service.

4 If the device in which this object is implemented has some sort of mechanism for changing connections, the state of these bits shall be changed to reflect the present state of the device.

5 Required if object supports any optional services. Optional if object does not support any optional services.

The additions made to this Object with revision 2 are:

Definition of Connection Status attribute for target-side, peer-to-peer connections

In the Connection Flags Attribute defined bits 4-6 for "T→O Real time transfer format". Changed the name of bits 1-3 from "Idle Style" to "O=→ Real time transfer format" and defined the value=3 for "Heartbeat"

Changed the Object Specific service Get_Status from optional to required, and changed the Set_Attribute_Single service from required to optional.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Added Instance attribute 11, Proxy Device ID. Updated the Get_Attributes_All response and Set_Attributes_all request to include attribute 11.

### 5B-11.1.1 Semantics

#### 5B-11.1.1.1 Revision – UINT data type

Defines the revision of the Connection Configuration Object Class Definition upon which the implementation is based. As time progressed it is possible for technical updates of the Connection Configuration Object Class Definition to occur for which an indication is desireable. This attribute enables these updates to be tracked inside devices.

The current value assigned to this attribute is three (3). If updates which require an increase in this value are made, then the value will be increased. Support of this attribute is required.

**Table 5B-11-2 Revision History**

| Revision | Reason for change and description of change from previous revision |
|---|---|
| 0x0001 | Initial Definition |
| 0x0002 | Added class attributes 33 and 34 |
| 0x0003 | Modification to make class attribute 5 conditional |
| 0x0004 – 0xFFFF | Reserved by CIP |

#### 5B-11.1.1.2 Format Number

The format value that shall be specified in the format field of each instance of attribute 9.

**Table 5B-11-3 Format Number Values**

| Values: |
|---|
| 0 – Single O→T/T→O tables, 16-bit words, 0-based offsets |
| 1 – Multiple O→T/T→O tables, 16-bit words, 0-based offsets |
| 2 - 99 – Reserved |
| 100 - 199 – Vendor Specfic |
| All other values reserved |

#### 5B-11.1.1.3 Edit Signature

Created and used by configuration software to detect modification to the instance attribute values. For ControlNet this value is initially 0 and is incremented each time a change complete operation is performed. For all other CIP networks this value is initially 0 and is set to a 32 bit CRC each time a change complete operation is performed (the polynomial is x^32+x^26+x^23+x^22+x^16+x^12+x^11+x^10+x^8+x^7+x^5+x^4+x^2+x^1+x^0), the CRC seed value shall be 0. The CRC is calculated on the set all attribute data stream for each connection instance (lowest to highest) plus class attribute 1, class attribute 2, class attribute 3 and class attribute 8.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 5B-11.2 Instance Attributes

**Table 5B-11-4 Connection Configuration Object Instance Attributes**

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 1 | Required | Get | V | Connection Status | Struct of | This attribute provides connection status. Only two bytes of extended status are included for error codes. | See Semantics section<br>Note: For access to complete connection status, it is recommended that attribute 23 (Full Connection Status) be implemented as well. |
| | | | | gen_status | USINT | General status | The General Status value returned in the Forward_Open/Large_Forward_Open response |
| | | | | reserved | USINT | Reserved | Shall be zero |
| | | | | ext_status | UINT | Extended status | If no extended status is returned, the ext_status shall be zero. |
| 2 | Required | Set | NV | Connection Flags | WORD | Connection flags | See Semantics |
| 3 | Required | Set | NV | Target Device ID | Struct of | | See Semantics section |
| | | | | vendor_id | UINT | Vendor ID | |
| | | | | product_type | UINT | Device Type | |
| | | | | product_code | UINT | Product Code | |
| | | | | major_rev | USINT | Major revision | |
| | | | | minor_rev | USINT | Minor revision | |
| 4 | Conditional | Set | NV | CS Data Index Number | UDINT | | See Semantics section |
| 5 | Required | Set | NV | Net Connection Parameters | Struct of | | |
| | | | | conn_timeout | USINT | Connection Timeout Multiplier | See Semantics section |
| | | | | xport_class_and_trig | BYTE | Transport Class and Trigger | See Semantics section |
| | | | | rpi_OT | UDINT | Originator to Target Requested Packet Interval | See Semantics section |
| | | | | net_OT | UINT | Originator to Target network connection parameters | See Semantics section |
| | | | | rpi_TO | UDINT | Target to Originator Requested Packet Interval | See Semantics section |
| | | | | net_TO | UINT | Target to Originator network connection parameters | See Semantics section |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Connection Configuration Object, Class Code: $F3_{Hex}$

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 6 | Required | Set | NV | Connection Path | Struct of | | |
| | | | | open_path_size | USINT | Open connection path size | See Semantics section |
| | | | | reserved | USINT | Reserved | |
| | | | | open_connection_path | Padded EPATH | Connection path | See Semantics section |
| 7 | Required | Set | NV | Proxy Config Data | Struct of | | See Semantics section |
| | | | | config_data_size | UINT | Length of config_data in bytes. | |
| | | | | config_data | Array of octet | Proxy Config data | |
| 8 | Required | Set | NV | Connection Name | STRING2 | User –assigned connection name encoded in UNICODE | See Semantics section |
| 9 | Required | Set | NV | I/O Mapping | Struct of | | See Semantics section |
| | | | | format_number | UINT | This number determines the format of this attribute. | The format value shall match the value of class attribute 8. |
| | | | | mapping_data_size | UINT | Size in bytes of the mapping_data field that follows. | |
| | | | | mapping_data | Array of BYTE | I/O mapping information associated with this instance. | |
| 10 | Required | Set | NV | Target Config Data | Struct of | | See Semantics section |
| | | | | config_data_size | UINT | Length of config_data in bytes. | |
| | | | | config_data | Array of octet | Target Config data | |
| 11 | Required | Set | NV | Proxy Device ID | Struct of | | See Semantics section |
| | | | | vendor_id | UINT | Vendor ID | |
| | | | | product_type | UINT | Device Type | |
| | | | | product_code | UINT | Product Code | |
| | | | | major_rev | USINT | Major revision | |
| | | | | minor_rev | USINT | Minor revision | |
| 12 | Conditional [1] | Set | NV | Connection Disable | BOOL | Indicates if this instance of the connection configuration object is disabled | 0 – This instance of the connection configuration object is enabled.<br>1 – This instance of the connection configuration object is disabled.<br>When an Open service is received this value shall be set to 0. When a Close or Stop service is received this value shall be set to 1.<br>The default value of this parameter is 0. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Connection Configuration Object, Class Code: $F3_{Hex}$

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|
| 13 | Conditional [2] | | | Safety Parameters | See CIP Safety Specification (Volume 5, Chapter 5) | | |
| 14 | Conditional [2] | | | Safety Connection Parameter CRC | | | |
| 15 | Conditional [2] | | | Safety Configuration Instance | | | |
| 16 | Conditional [2] | | | Safety ID Allocation | | | |
| 17 | Conditional [2] | | | Safety Target Connection Serial Number | | | |
| 18 | Optional | Set | | Net Connection Parameters Attribute Selection | USINT | Selects between attribute 5 and attribute 19 | See Semantics section |
| 19 | Conditional [3] | Set | | Large Net Connection Parameters | Struct of | | |
| | | | | conn_timeout | USINT | Connection Timeout Multiplier | See Semantics section |
| | | | | xport_class_and_trigger | BYTE | Transport Class and Trigger | See Semantics section |
| | | | | rpi_OT | UDINT | Originator to Target Requested Packet Interval | See Semantics section |
| | | | | net_OT | UDINT | Originator to Target large network connection parameters | See Semantics section |
| | | | | rpi_TO | UDINT | Target to Originator Requested Packet Interval | See Semantics section |
| | | | | net_TO | UDINT | Target to Originator large network connection parameters | See Semantics section |
| 20 | Conditional [2] | | | Format Type | See CIP Safety Specification (Volume 5, Chapter 5) | | |
| 21 | Conditional [2] | | | Format Status | | | |
| 22 | Conditional [2] | | | Max Fault Number | | | |
| 23 | Optional | Get | V | Full Connection Status | Struct of | This attribute is the full connection status (handles extended status longer than a single WORD) | |
| | | | | gen_status | USINT | General status | The General Status value returned in the Forward_Open/Large_Forward_Open response |
| | | | | size of ext_status | USINT | Number of 16-bit words in ext_status array. | |
| | | | | ext_status | Array of WORD | Extended status [4] | Empty or the Extended Status value returned in the Forward_Open/Large_Forward_Open response |
| 24 | Optional | Set | NV | Request Priority and Timeout | STRUCT of: | See section 3 5.5.1.4, Unconnected Request Timing. Default timeout value is vendor specific, 5 seconds (0x05, 156) recommended. | |
| | | | | Priority/Time_tick [5] | BYTE | | |
| | | | | Time-out_ticks [5] | USINT | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attr ID | Need in Implem | Access Rule | NV | Name | Data Type | Description of Attribute | Semantics of Values |
|---|---|---|---|---|---|---|---|

Table Footnotes:

1 Attribute 12 shall be supported if conditional services Open_Connection (4C) and Close_Connection (4D) are supported. Attribute 12 shall not be supported if conditional services Open_Connection (4C) and Close_Connection (4D) are not supported.

2 These attributes are not allowed if the device is not a safety device. If the device is a safety device, see Volume 5, CIP Safety.

3 Conditional attribute 19 is required if attribute 18 is supported, otherwise conditional attribute 19 is not allowed.

4 If size_of_ext_status is non-zero, then the first word of ext_status shall be the same value as the ext_status element in the Connection Status structure of attribute 1.

5 This attribute is not included in the Get_Attributes_All response or Set_Attributes_All request.

### 5B-11.2.1 I/O Mapping Formats

The following tables describe the structure of the mapping_data field of instance attribute 9 for each of the format numbers.

**Table 5B-11-5 Mapping Formats**

| Format Number | Mapping_data_size (in bytes) | Name | Data Type | Description |
|---|---|---|---|---|
| 0 | 4 | Single I/O tables | Struct of: | 16 bit words, 0-based 16-bit word offsets |
| | | O→T offset | UINT | Offset into O→T image table. |
| | | T→O offset | UINT | Offset into T→O image table. |
| 1 | 8 | Multiple I/O tables | Struct of: | Table selection, 16 bit words, 0-based 16-bit word offsets |
| | | O→T table | UINT | O→T image table selection |
| | | O→T offset | UINT | Offset into O→T image table. |
| | | T→O table | UINT | T→O image table selection |
| | | T→O offset | UINT | Offset into T→O image table. |

### 5B-11.2.2 Semantics

#### 5B-11.2.2.1 Connection Status – Attribute 1

**Table 5B-11-6 Originator Connection Status Values**

| General Status | Extended Status | Description |
|---|---|---|
| 0x00 | n/a | Connection is open |
| 0x01 | 0x0000 or greater | See the table of Connection Manager Service Request Error Codes in Chapter 3, in the Error Code Listing subsection |
| 0x02 – 0x26 | 0x0000 or greater | See General Status codes in Appendix B |
| 0xD0 | 0x0001 | Connection is closed or stopped (via the Close or Stop services) |
| | 0x0002 | Connection open is pending |
| | 0x0003 | Connection close is pending |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-11-7 Target Connection Status Values**

| General Status | Extended Status | Description |
|---|---|---|
| 0x00 | 0x0000 | Connection is not open |
| 0x00 | 0x0001 or greater | The number of connections to this target |
| 0xD0 | 0x0001 | Connection is closed or stopped (via the Close or Stop services) |

### 5B-11.2.2.2 Connection Flags – Attribute 2

**Table 5B-11-8 Connection Flags Attribute Definition**

| Bit | Meaning |
|---|---|
| 0 | Connection<br>0 = Originator<br>1 = Target |
| 1 – 3 | O→T Real time transfer format, see section 3-6.1<br>000 – 32-bit header format includes run/idle information.<br>001 – Zero length data format indicates idle.<br>010 – Modeless format – no run/idle notification.<br>011 – Heartbeat format – no run/idle notification.<br>100 – Reserved.<br>101 – Safety format (See Volume 5)<br>100 thru 111 – reserved for future use. |
| 4 - 6 | T→O Real time transfer format, see section 3-6.1<br>000 – 32-bit header format includes run/idle information.<br>001 – Zero length data format indicates idle.<br>010 – Modeless format – no run/idle notification.<br>011 – Heartbeat format – no run/idle notification.<br>100 – Reserved.<br>101 – Safety format (See Volume 5)<br>100 thru 111 – reserved for future use. |
| 7 - 15 | Reserved |

### 5B-11.2.2.3 Target Device ID – Attribute 3

This attribute is the identity of the target device for this connection instance. This identity information is not used for verifying (keying) the online target device, it is used by a configuration tool to locate the correct electronic data sheet for the connection configuration described in this CCO instance. For Target instances this attribute shall be the identity of the device containing this CCO object.

### 5B-11.2.2.4 CS Data Index Number – Attribute 4

The CS Data Index Number attribute is required for a device on ControlNet; otherwise this attribute shall not be implemented.

The CS Data Index Number is a value set by the configuration software. This is the connection_index value returned from the Schedule Object read service (See Volume 4, ControlNet Adaptation of CIP, Chapter 5 for the definition of the Scheduling Object services). This attribute is ignored for target instances.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-11.2.2.5 Net Connection Parameters – Attribute 5

#### 5B-11.2.2.5.1 conn_timeout

The conn_timeout is the value used for the connection timeout multiplier field defined in Chapter 3 for a description of the Forward_Open service request. conn_timeout is ignored for target instances.

#### 5B-11.2.2.5.2 xport_class_and_trig

The xport_class_and_trig is the value used for the Transport Type/Trigger field in the Chapter 3 definition of the Forward_Open service request. For target instances the xport_class_and_trig field shall be ignored. The device containing this CCO object shall determine if the similar Transport Type/Tirgger parameter of the Forward_Open service request is supported as specified in section 3-5.4.1.13 (Transport Class and Trigger).

#### 5B-11.2.2.5.3 rpi_OT

The rpi_OT is the value used for the O→T RPI field in the Chapter 3 definition of the Forward_Open service request. rpi_OT is ignored for target instances.

#### 5B-11.2.2.5.4 net_OT

The net_OT is the value used for the O→T Network Connection Parameters field in the Chapter 3 definition of the Forward_Open service request.

For target instances, the net_OT field shall be ignored, with the exception of the connection size. The device containing this CCO object shall determine if the similar O→T Network Connection Parameters parameter of the Forward_Open service request is supported as specified in section 3-5.4.1.1 (Network Connection Parameters).

#### 5B-11.2.2.5.5 rpi_TO

The rpi_TO is the value used for the T→O RPI field in the Chapter 3 definition of the Forward_Open service request). rpi_TO is ignored for target instances.

#### 5B-11.2.2.5.6 net_TO

The net_TO is the value used for the T→O Network Connection Parameters field in the Chapter 3 definition of the Forward_Open service request.

For target instances, the net_TO field shall be ignored with the exception of the connection size. The device containing this CCO object shall determine if the similar T→O Network Connection Parameters parameter of the Forward_Open service request is supported as specified in section 3-5.4.1.1 (Network Connection Parameters).

### 5B-11.2.2.6 Connection Path – Attribute 6

#### 5B-11.2.2.6.1 open_path_size

The open_path_size is the value used for the Connection_Path_Size field in the Chapter 3 definition of the Forward_Open/Large_Forward_Open service requests. For target instances the open_path_size is used for matching the Connection_Path_Size received in a Forward_Open/Large_Forward_Open serivce request.

#### 5B-11.2.2.6.2 open_connection_path

The open_connection_path is the value used for the Connection_Path field in the Chapter 3 definition of the Forward_Open/Large_Forward_Open service request. For target instances the open_connection_path is used for matching the Connection_Path parameter received in a Forward_Open/Large_Forward_Open service request.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 5B-11.2.2.7 Proxy Config Data – Attribute 7

This does not apply to target instances. The data specified in attributes 7 and 10 are concatenated (in that order) into a single data segment, then appended to the connection path in attribute 6 to form the complete path sent in the Forward_Open/Large_Forward_Open services of the Connection Manager object.

The configuration data may be split between attributes 7 and 10. When a connection is a non-proxied connection the convention that shall be followed is that all configuration data is placed in attribute 10 (Target Config). When a connection is a proxied connection the convention that shall be followed is that any configuration data intended for the proxying device is placed in attribute 7 (Proxy Config) and any configuration data intended for the proxied device is placed in attribute 10 (Target Config). For connections to devices that are configured using the target device EDS this information comes from the Connection Manager section of the EDS, fields 9 and 10 of the Connection entry (see the Connection Manager Section of Chapter 7 for more information). Proxy Config data is ignored for target instances.

#### 5B-11.2.2.8 Connection Name – Attribute 8

This Connection Name field allows a user to name each connection instance. The connection name has no meaning to the Connection Configuration Object. The Connection Name shall not exceed 255 characters in length. See Character String Data Types (section C-2.1.2) for details on the encoding for the STRING2 data type.

#### 5B-11.2.2.9 I/O Mapping Attribute – Attribute 9

The I/O Mapping attribute contains I/O mapping data. The I/O mapping data specifies image table locations where target to originator data is placed and where originator to target data is obtained.

#### 5B-11.2.2.10 Target Config Data – Attribute 10

This note does not apply to target instances. The data specified in attributes 7 and 10 are concatenated (in that order) into a single data segment, then appended to the connection path in attribute 6 to form the complete path sent in the Forward Open service of the Connection Manager object. All the configuration data may be placed in attribute 7 or all the configuration data may be placed in attribute 10 or the configuration data may be split between attributes 7 and 10. When a connection is a proxied connection the convention that shall be followed is that any configuration data intended for the proxying device is placed in attribute 7 and any configuration data intended for the proxied device is placed in attribute 10. For connections to devices that are configured using the target device EDS this information comes from the Connection Manager section of the EDS, fields 11 and 12 of the Connection entry (see the Connection Manager Section of Chapter 7 for more information). Target Config data is ignored for target instances.

#### 5B-11.2.2.11 Proxy Device ID – Attribute 11

This attribute is the identity of the device proxying for the target device for this connection instance. This identity information is not used for verifying (keying) the online target device, it is used by a configuration tool to locate the correct electronic data sheet for the connection configuration described in this CCO instance. For Target instances and Connections that are not proxied this attribute shall be ignored.

#### 5B-11.2.2.12 Net Connection Parameters Attribute Selection – Attribute 18

The Net Connection Parameters Attribute Selection attribute selects between the use of instance attribute 5 (Net Connection Parameters) and attribute 19 (Large Net Connection Parameters). The valid values are:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

0 = indicates instance attribute 5 shall be used (default)

1 = indicates instance attribute 19 shall be used

2-255 = reserved

If this optional attribute is not supported attribute 5 shall be used.

#### 5B-11.2.2.13 Large Net Connection Parameters – Attribute 19

##### 5B-11.2.2.13.1 conn_timeout

The conn_timeout is the value used for the connection timeout multiplier field defined in Chapter 3 for a description of the Large_Forward_Open service request. conn_timeout is ignored for target instances.

##### 5B-11.2.2.13.2 xport_class_and_trig

The xport_class_and_trig is the value used for the Transport Type/Trigger field in the Chapter 3 definition of the Large_Forward_Open service request. For target instances the xport_class and_trig field shall be ignored. The device containing this CCO object shall determine if the similar Transport Type/Trigger parameter of the Large_Forward_Open service request is supported as specified in section 3-5.4.1.13 (Transport Class and Trigger).

##### 5B-11.2.2.13.3 rpi_OT

The rpi_OT is the value used for the O→T RPI field in the Chapter 3 definition of the Large_Forward_Open service request. rpi_OT is ignored for target instances.

##### 5B-11.2.2.13.4 net_OT

The net_OT is the value used for the O→T Network Connection Parameters field in the Chapter 3 definition of the Large_Forward_Open service request.

For target instances, the net_OT field shall be ignored, with the exception of the connection size. The device containing this CCO object shall determine if the similar O→T Network Connection Parameters parameter of the Large_Forward_Open service request is supported as specified in section 3-5.4.1.1 (Network Connection Parameters).

##### 5B-11.2.2.13.5 rpi_TO

The rpi_TO is the value used for the T→O RPI field in the Chapter 3 definition of the Large_Forward_Open service request. rpi_TO is ignored for target instances.

##### 5B-11.2.2.13.6 net_TO

The net_TO is the value used for the T→O Network Connection Parameters field in the Chapter 3 definition of the Large_Forward_Open service request.

For target instances, the net_TO field shall be ignored, with the exception of the connection size. The device containing this CCO object shall determine if the T→O Network Connection Parameters parameter of the Large_Forward_Open service request is supported as specified in section 3-5.4.1.1 (Network Connection Parameters).

### 5B-11.3 Connection Configuration Object Change Control

Changes to Connection Configuration object attributes can only be made after a Change Start service has been successfully issued. Changes to Connection Configuration object attributes are applied after a Change Complete service has been successfully issued.

The following services are valid during a change operation. A change operation is started by issuing a Change Start service and completed by issuing a Change Complete service:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-11-9 Services Valid During a Change Operation**

| Service Code | Service Name |
|---|---|
| 02hex | Set_Attributes_All |
| 08hex | Create |
| 09hex | Delete |
| 10hex | Set_Attribute_Single |
| 15hex | Restore |
| 4Bhex | Kick Timer |
| 51hex | Change Complete |
| 52hex | Audit Changes |

## 5B-11.4 Common Services

The Connection Configuration Object provides the following Common Services:

**Table 5B-11-10 Connection Configuration Object Common Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 01hex | Required | Required | Get_Attributes_All | Gets all attributes of the specified instance. |
| 02hex | N/A | Required | Set_Attributes_All | Sets all attributes of the specified instance. |
| 08hex | Required | N/A | Create | Creates a new connection instance. |
| 09hex | Required | Required | Delete | Deletes an existing connection instance. |
| 0Ehex | Optional | Required | Get_Attribute_Single | Returns the contents of the specified attribute. |
| 10hex | Required | Optional | Set_Attribute_Single | Modifies an attribute value. |
| 15hex | Required | Required | Restore | Restore current connection attributes. |
| See Appendix A for description of these services | | | | |

### 5B-11.4.1 Get_Attributes_All (Service Code 0x01)

At the **Class level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:

**Table 5B-11-11 Get_Attributes_All Response Data – Class Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | UINT | Revision | 1 [1] |
| 2 | UDINT | Max Instance | |
| 3 | UDINT | Number of Instances | |
| 8 | UINT | Format Number | |
| 9 | UDINT | Edit Signature | |

Table Footnotes:

1. The default value only applies for Revision 1 of this object. Newer revisions require that this attribute be implemented, which will override this default value.

At the **Instance level**, the order of the attributes returned in the Attribute Values portion of the Get_Attributes_All response is as follows:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-11-12 Get_Attributes_All Response Data – Instance Level**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 1 | STRUCT of: | Connection Status | |
| | USINT | gen_status | |
| | USINT | reserved | |
| | UINT | ext_status | |
| 2 | WORD | Connection Flags | |
| 3 | STRUCT of: | Target Device ID | |
| | UINT | Vendor ID | |
| | UINT | Device Type | |
| | UINT | Product Code | |
| | USINT | Major Revision | |
| | USINT | Minor Revision | |
| 4 | UDINT | CS Data Index Number | 0xFFFFFFFF |
| 5 | STRUCT of: | Net Connection Parameters | |
| | USINT | conn_timeout | |
| | BYTE | xport_class_and_trig | |
| | UDINT | rpi_OT | |
| | UINT | net_OT | |
| | UDINT | rpi_TO | |
| | UINT | net_TO | |
| 6 | STRUCT of: | Connection Path | |
| | USINT | open_path_size | |
| | USINT | reserved[1] | |
| | Padded EPATH | open_connection_path | |
| 7 | STRUCT of: | Proxy Config Data | |
| | UINT | config_data_size | |
| | ARRAY of octet | config_data | |
| – | octet | Pad[2] | |
| 10 | STRUCT of: | Target Config Data | |
| | UINT | config_data_size | |
| | ARRAY of octet | config_data | |
| – | Octet | Pad[3] | |
| 8 | STRING2 | Connection Name | |
| 9 | STRUCT of: | I/O Mapping | |
| | UINT | format_number | |
| | UINT | mapping_data_size | |
| | ARRAY of BYTE | mapping_data | |
| – | octet | Pad[4] | |
| 11 | STRUCT of: | Proxy Device ID | |
| | UINT | Vendor ID | |
| | UINT | Device Type | |
| | UINT | Product Code | |
| | USINT | Major Revision | |
| | USINT | Minor Revision | |
| 13-17 | See Safety Volume (Volume 5, Chapter 5)<br><br>55 octets total | Safety Parameters | (null) [5] |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 12 | BOOL[6] | Connection Disable | 0 |
| 18 | USINT | Net Connection Parameters Attribute Selection | 0 |
| 19 | STRUCT of: | Large Net Connection Parameters | |
| | USINT | conn_timeout | 0 |
| | BYTE | xport_class_and_trigger | 0 |
| | UDINT | rpi_OT | 0 |
| | UDINT | net_OT | 0 |
| | UDINT | rpi_TO | 0 |
| | UDINT | net_TO | 0 |
| 20-22 | See Safety Volume (Volume 5, Chapter 5)<br><br>Variable Length. | Additional Safety Parameters | (null) [5] |

Table Footnotes:

1 This pad byte shall be zero.

2 This pad byte shall only be included if the length of the Proxy Config Data (Attribute #7) is an odd number of bytes in length.

3 This pad byte shall only be included if the length of the Target Config Data (Attribute #10) is an odd number of bytes in length.

4 This pad byte shall only be included if the length of the I/O Mapping (Attribute #9) is an odd number of bytes in length.

5 The Safety Parameters (55 bytes) and Additional Safety Parameters (variable length) blocks shall only be included in the response if either the O→T or T→O real time transfer format in the Connection Flags (Attribute #2) indicates the format is Safety.

6 The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.

**Important:** A device is only required to include data in the response data array up to the last implemented attribute as defined in Appendix A-4.1, Get_Attributes_All.

### 5B-11.4.2 Set_Attributes_All (Service Code 0x02)

This service modifies the contents of the attributes shown in Table 5B-11.14 and supports the following error codes in addition to other error codes listed in Appendix B:

**Table 5B-11-13 Set_Attributes_All Service Request Errors**

| General Status | Extended Status | Error Name | Description |
|---|---|---|---|
| 0x0C | None | Object State Conflict | The Connection Configuration Object cannot accept the Set_Attribute_Single service when an edit session is not active |

The structure of the Set_Attributes_All request is shown in the table below.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-11-14 Set_Attributes_All Request Data**

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 2 | WORD | Connection Flags | |
| 3 | STRUCT of: | Target Device ID | |
| | UINT | Vendor ID | |
| | UINT | Device Type | |
| | UINT | Product Code | |
| | USINT | Major Revision | |
| | USINT | Minor Revision | |
| 4 | UDINT | CS Data Index Number | 0xFFFFFFFF |
| 5 | STRUCT of: | Net Connection Parameters | |
| | USINT | conn_timeout | |
| | BYTE | xport_class_and_trig | |
| | UDINT | rpi_OT | |
| | UINT | net_OT | |
| | UDINT | rpi_TO | |
| | UINT | net_TO | |
| 6 | STRUCT of: | Connection Path | |
| | USINT | open_path_size | |
| | USINT | reserved[1] | |
| | Padded EPATH | open_connection_path | |
| 7 | STRUCT of: | Proxy Config Data | |
| | UINT | config_data_size | |
| | ARRAY of octet | config_data | |
| – | octet | Pad[2] | |
| 10 | STRUCT of: | Target Config Data | |
| | UINT | config_data_size | |
| | ARRAY of octet | config_data | |
| – | octet | Pad[3] | |
| 8 | STRING2 | Connection Name | |
| 9 | STRUCT of: | I/O Mapping | |
| | UINT | format_number | |
| | UINT | mapping_data_size | |
| | ARRAY of BYTE | mapping_data | |
| – | octet | Pad[4] | |
| 11 | STRUCT of: | Proxy Device ID | |
| | UINT | Vendor ID | |
| | UINT | Device Type | |
| | UINT | Product Code | |
| | USINT | Major Revision | |
| | USINT | Minor Revision | |
| 13, 15, 16 | See Safety Volume (Volume 5, Chapter 5)<br><br>49 octets total | Safety Parameters | (null) [5] |
| 12 | BOOL[6] | Connection Disable | 0 |
| 18 | USINT | Net Connection Parameters Attribute Selection | 0 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attribute ID | Data Type | Attribute Name | Default Value (if not implemented) |
|---|---|---|---|
| 19 | STRUCT of: | Large Net Connection Parameters | |
| | USINT | conn_timeout | 0 |
| | BYTE | xport_class_and_trigger | 0 |
| | UDINT | rpi_OT | 0 |
| | UDINT | net_OT | 0 |
| | UDINT | rpi_TO | 0 |
| | UDINT | net_TO | 0 |
| 20, 22 | See Safety Volume (Volume 5, Chapter 5)<br><br>Variable Length. | Additional Safety Parameters | (null) [5] |

Table Footnotes:
1 This pad byte shall be zero.
2 This pad byte shall only be included if the length of the Proxy Config Data (Attribute #7) is an odd number of bytes in length.
3 This pad byte shall only be included if the length of the Target Config Data (Attribute #10) is an odd number of bytes in length.
4 This pad byte shall only be included if the length of the I/O Mapping (Attribute #9) is an odd number of bytes in length.
5 The Safety Parameters (49 bytes) and Additional Safety Parameters (variable length) blocks shall only be included in the request if either the O→T or T→O real time transfer format in the Connection Flags (Attribute #2) indicates the format is Safety.
6 The BOOL type shall be encoded in bit 0 of a single octet. Bits 1-7 shall be zero, as defined in section C-5.2.1.

Data for unsupported attributes shall be accepted if the default values are sent. The service shall be rejected if non-default values are sent for unsupported attributes.

### 5B-11.4.3 Create (Service Code 0x08)

This service creates a new instance of a Connection Configuration object. Initial attribute values may also be specified with this service. The created instance number is assigned by the class and returned to the requestor. The following request parameters are defined for this service:

**Table 5B-11-15 Create Service Request**

| Parameter | Data Type | Description |
|---|---|---|
| Connection Flags | WORD | Connection flags |
| NumConnParams | UINT | Number of attribute/value pairs that follow. |
| ConnParams | Array of Struct of | List of attribute number/value pairs |
| | UINT | Attribute number |
| | Object/class attribute-specific Struct | Attribute value |

Data for unsupported attributes shall be accepted if the default values are sent. The service shall be rejected if non-default values are sent for unsupported attributes.

This service supports the following error codes in addition to other error codes listed in Appendix B:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-11-16 Create Service Request Errors**

| General Status | Extended Status | Error Name | Description |
|---|---|---|---|
| 0x02 | 0x0001 | Resource unavailable | The maximum number of instances already exist |
| | 0x0002 | Resource unavailable | Not enough memory on device |
| 0x03 | None | Invalid parameter value | The attribute count is invalid |
| 0x08 | None | Service not supported | Unimplemented service |
| 0x0C | None | Object State conflict | The Connection Configuraton Object cannot execute the Set_Attribute_Single service when an edit session is not active |
| 0x0E | None | Attribute not settable | Attempt to set a read-only attribute |
| 0x13 | None | Not enough data | The request was too short or truncated |
| 0x1C | None | Missing attribute list entry data | The required attribute was missing |

### 5B-11.4.4 Delete (Service Code 0x09)

This service deletes existing connection instances. If addressed to the class-level, all connection instances are deleted. If addressed to the instance-level, only the addressed instance is deleted. This service supports the following error codes in addition to other codes listed in Appendix B:

**Table 5B-11-17 Delete Service Request Errors**

| General Status | Extended Status | Error Name | Description |
|---|---|---|---|
| 0x0C | None | Object State conflict | The Connection Configuration Object cannot execute the Delete service when an edit session is not active |

### 5B-11.4.5 Restore (Service Code 0x15)

This service shall discard modifications to instances that have not yet been committed by the Change_Complete service. If the Restore service is addressed to the class (instance 0), then pending modifications for all instances shall be discarded and the edit session shall be ended. If addressed to a specific instance, only modifications for that instance shall be discarded and the edit session shall not be ended.

This service supports the following error codes in addition to other error codes listed in Appendix B:

**Table 5B-11-18 Restore Service Request Errors**

| General Status | Extended Status | Error Name | Description |
|---|---|---|---|
| 0x0C | None | Object State conflict | The Connection Configuration Object cannot execute the Restore service when an edit session is not active |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-11.4.6 Object-specific Services

The Connection Configuration Object provides the following Object Specific Services:

**Table 5B-11-19 Connection Configuration Object Object-specific Services**

| Service Code | Need in Implementation | | Service Name | Description of Service |
|---|---|---|---|---|
| | Class | Instance | | |
| 4Bhex | Required | N/A | Kick_Timer | Kicks Edit Watchdog Timer |
| 4Chex | Conditional* | Conditional* | Open_Connection | Opens connections |
| 4Dhex | Conditional* | Conditional* | Close_Connection | Closes connections |
| 4Ehex | Conditional* | Conditional* | Stop_Connection | Stops connections |
| 4Fhex | Required | N/A | Change_Start | Manages session editing |
| 50hex | Required | N/A | Get_Status | Get status for multiple connections |
| 51hex | Required | N/A | Change_Complete | Completes session editing |
| 52hex | Required | N/A | Audit_Changes | Audits pending changes |

*The Open_Connection and Close_Connection services shall either both be supported or neither the Open_Connection or Close_Connection services shall be supported. The Stop_Connection service may only be supported if the Open_Connection service is supported

### 5B-11.4.7 Kick_Timer (Service Code 0x4B)

This service shall reinitialize the edit watchdog timer. Upon successful execution of the Change_Start service, the edit watchdog timer shall be started with a period of 60 seconds. This timer is used to recover from the loss of a configuration client between the Change_Start and Change_Complete/Restore operations. Receipt of any service request shall reset the edit watchdog timer. Clients may request this service to reset the timer without otherwise affecting the state of the Connection Configuration object. If the edit watchdog timer expires, all pending modifications shall be discarded and the edit session shall be ended.

### 5B-11.4.8 Open_Connection (Service Code 0x4C)

The Open_Connection service shall cause the connection associated with an instance of the Connection Configuration object to open. If the Open_Connection service is addressed to the class (instance 0), then all connection instances shall be opened. If addressed to a specific instance, then only that connection instance shall be opened.

### 5B-11.4.9 Close_Connection (Service Code 0x4D)

The Close_Connection service shall cause the connection associated with an instance of the Connection Configuration object to close. If the Close_Connection service is addressed to the class (instance 0), then all connection instances shall be closed. If addressed to the instance level, then only the specified instance shall be closed. The Close_Connection service shall initiate a “graceful” connection shutdown, that is, a Forward_Close request shall be sent to the connection target. Once a connection has been closed by this service it shall remain closed until an Open_Connection service is issued.

### 5B-11.4.10 Stop_Connection (Service Code 0x4E)

The Stop_Connection service shall cause the connection associated with an instance of the Connection Configuration object to stop producing data immediately without sending a Forward_Close request to the connection target. If the Stop_Connection service is addressed to the class (instance 0), then all connection instances shall be stopped. If addressed to the instance level, then the specified instance shall be stopped. Once a connection has been stopped, it remains stopped until a subsequent Open_Connection service request is issued.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 5B-11.4.11 Change_Start (Service Code 0x4F)

This service shall:

- signal the beginning of an edit session;
- synchronize the current and pending attributes;
- place all connections in the "changeable" state;
- start the edit watchdog timer.

Change_Start shall be requested prior to performing any services that modify the attributes of a connection. This service shall only be addressed to the class-level (instance 0). If a Change_Start service is received while an edit session is active, an Object State Conflict error (0x0C) shall be returned.

This service supports the following error codes in addition to other error codes listed in Appendix B:

**Table 5B-11-20 Change_Start Service Request Error Codes**

| General Status | Extended Status | Error Name | Description |
|---|---|---|---|
| 0x0C | None | Object State conflict | An edit session is already active |
| 0x10 | None | Device State conflict | The device is in a state that prevents starting an edit session |

### 5B-11.4.12 Get_Status (Service Code 0x50)

The Get_Status service shall retrieve the status attribute (attribute 1) for multiple connections via a single transaction. This service shall be supported at the class-level (instance 0) only. Given a starting instance number, the Get_Status service shall return instance/status pairs until either the response buffer is full or the status of all connections has been returned.

The following request parameter is defined:

**Table 5B-11-21 Get_Status Service Parameter**

| Parameter | Data Type | Description |
|---|---|---|
| Starting Instance | UDINT | Starting instance number |

The following response parameters are defined for this service:

**Table 5B-11-22 Get_Status Service Response**

| Parameter | Data Type | Description |
|---|---|---|
| Done Indicator | UINT | 0 = More status to be retrieved<br>1 = All connection status information has been retrieved.<br>All other values reserved. |
| NumStatusEntries | UINT | Number of instance/status pairs that follow. |
| StatusEntries | Array of Struct of | List of instance/status pairs |
| | UDINT | Connection Configuration instance number |
| | USINT | General status |
| | USINT | Reserved, shall be 0 |
| | UINT | Extended status |

This service supports the following error codes in addition to other error codes listed in Appendix B:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-11-23 Get_Status Service Errors**

| General Status | Extended Status | Error Name | Description |
|---|---|---|---|
| 0x03 | None | Invalid Parameter Value | The attribute count is invalid |

### 5B-11.4.13 Change_Complete (Service Code 0x51)

This service shall signal the completion of an edit session. Pending attributes for all modified connection instances shall be applied. This service shall take a parameter indicating the type of change being performed; either full or incremental. If an incremental edit is specified, then only the connections that have been modified shall be broken and re-established. If a full edit is specified, then all connections shall be broken and all supporting resources shall be freed and reallocated before attempting to re-establish the connections. The Change_Complete service shall be supported at the class-level (instance 0) only.

If optional instance attribute 12 is supported and the value of instance attribute 12 is FALSE or if optional instance attribute 12 is not supported an attempt to open the connection shall be made. If optional instance attribute 12 is supported and the value of instance attribute 12 is TRUE, no attempt shall be made to open the connection.

The following request parameter is defined:

**Table 5B-11-24 Change_Complete Service Parameter**

| Parameter | Data Type | Description |
|---|---|---|
| Change Type | UINT | 0 = Full<br>1 = Incremental<br>All other values reserved. |

This service supports the following error codes in addition to other error codes listed in Appendix B:

**Table 5B-11-25 Change_Complete Service Errors**

| General Status | Extended Status | Error Name | Description |
|---|---|---|---|
| 0x02 | None | Resource unavailable | Indicates that there is not enough memory on the host device to support the specified configuration. |
| 0x0C | None | Object State conflict | An edit session is not active |
| 0x10 | None | Device state conflict | The device is in a state that prevents completing an edition session |

### 5B-11.4.14 Audit_Changes (Service Code 0x52)

This service shall verify whether or not there is enough memory on the host device to support a proposed configuration. Like the Change_Complete service, this service shall take a parameter indicating the type of change being performed; either full or incremental. The Audit_Changes service shall be supported at the class-level (instance 0) only. This service allows a configuration client to determine if all pending changes will be successful before actually committing the changes with the Change_Complete service.

The following request parameter is defined:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 5B-11-26 Audit_Changes Service Parameter**

| Parameter | Data Type | Description |
|---|---|---|
| Change Type | UINT | 0 = Full<br>1 = Incremental<br>All other values reserved. |

This service supports the following error codes in addition to other error codes listed in Appendix B:

**Table 5B-11-27 Audit_Changes Service Errors**

| General Status | Extended Status | Error Name | Description |
|---|---|---|---|
| 0x02 | None | Resource unavailable | Indicates that there is not enough memory on the host device to support the specified configuration. |
| 0x0C | None | Object State conflict | An edit session is not active |
| 0x10 | None | Device state conflict | The device is in a state that prevents completing an audit |

## 5B-11.5 Behavior

The flowchart below summarizes the process of creating, editing, and deleting connections.

**Figure 5B-11.1 Connection Configuration Object Edit Flowchart**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Volume 1: Common Industrial Protocol (CIP™) Specification

# Chapter 6: Device Profiles


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Contents


6-1 Introduction ..... 10
6-2 The Object Model ..... 10
6-2.1 All Objects Present in a Device ..... 11
6-2.2 Objects That Affect Behavior ..... 13
6-2.3 Object Interfaces ..... 13
6-3 I/O Data Format ..... 13
6-3.1 I/O Assembly Instances ..... 14
6-3.2 Format of I/O Assembly Data Attribute ..... 14
6-3.3 Map of I/O Assembly Data Attribute Components ..... 14
6-4 Device Configuration ..... 15
6-4.1 Parameter Data ..... 16
6-4.2 Effect of Configuration Parameters on Behavior ..... 16
6-4.3 Parameter Groups ..... 16
6-4.4 Public Interfaces to Device Configuration Data ..... 16
6-4.4.1 Parameter Object ..... 16
6-4.4.2 Configuration Assembly Object ..... 17
6-5 Extended Device Profiles ..... 17
6-6 Device Profile Numbering Scheme ..... 18
6-7 Device Profiles ..... 19
6-8 Generic Device ..... 22
6-8.1 Object Model ..... 22
6-8.2 How Objects Affect Behavior ..... 23
6-8.3 Defining Object Interfaces ..... 23
6-9 Limit Switch Device ..... 24
6-9.1 Object Model ..... 24
6-9.2 How Objects Affect Behavior ..... 25
6-9.3 Defining Object Interfaces ..... 25
6-9.4 I/O Assembly Instances ..... 25
6-9.5 I/O Assembly Data Attribute Format ..... 25
6-9.6 Mapping I/O Assembly Data Attribute Components ..... 25
6-9.7 Defining Device Configuration ..... 26
6-9.7.1 Parameter Object Instances ..... 26
6-9.7.2 Mapping Parameter Object Data ..... 26
6-9.7.3 Configuration Parameter Definitions ..... 26
6-9.8 Effect of Configuration Parameters on Behavior ..... 26
6-10 Inductive Proximity Switch Device ..... 28
6-10.1 Object Model ..... 28
6-10.2 How Objects Affect Behavior ..... 29
6-10.3 Defining Object Interfaces ..... 29
6-10.4 I/O Assembly Instances ..... 29
6-10.5 I/O Assembly Data Attribute Format ..... 29
6-10.6 Mapping I/O Assembly Data Attribute Components ..... 29
6-10.7 Defining Device Configuration ..... 30
6-10.7.1 Parameter Object Instances ..... 30
6-10.7.2 Mapping Parameter Object Data ..... 30
6-10.7.3 Configuration Parameter Definitions ..... 30
6-10.8 Effect of Configuration Parameters on Behavior ..... 30
6-11 Photoelectric Sensor Device ..... 32
6-11.1 Object Model ..... 32
6-11.2 How Objects Affect Behavior ..... 33
6-11.3 Defining Object Interfaces ..... 33
6-11.4 I/O Assembly Instances ..... 33
6-11.5 I/O Assembly Data Attribute Format ..... 33



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

6-11.6 Mapping I/O Assembly Data Attribute Components ..... 33
6-11.7 Defining Device Configuration ..... 33
6-11.7.1 Parameter Object Instances ..... 34
6-11.7.2 Mapping Parameter Object Data ..... 34
6-11.7.3 Configuration Parameter Definitions ..... 34
6-11.8 Effect of Configuration Parameters on Behavior ..... 34
6-12 General Purpose Discrete I/O Device ..... 36
6-12.1 Object Model ..... 36
6-12.2 How Objects Affect Behavior ..... 38
6-12.3 Defining Object Interfaces ..... 38
6-12.4 I/O Assembly Instances ..... 38
6-12.5 I/O Assembly Data Attribute Format ..... 40
6-12.6 Mapping I/O Assembly Data Attribute Components ..... 47
6-12.7 Defining Device Configuration ..... 48
6-12.7.1 Input Configuration ..... 48
6-12.8 Output Configuration Assembly Instances ..... 48
6-12.9 Output Configuration Assembly Data Attribute Format ..... 48
6-12.10 Mapping Output Configuration Assembly Data Attribute Components ..... 48
6-13 Communications Adapter Device ..... 50
6-13.1 Object Model ..... 50
6-14 General Purpose Analog I/O Device ..... 52
6-15 AC/DC Drive Devices ..... 53
6-15.1 Multiple axes on one drive ..... 53
6-15.2 Object Model ..... 53
6-15.3 How Objects Affect Behavior ..... 56
6-15.4 Defining Object Interfaces ..... 56
6-15.5 I/O Assembly Instances ..... 56
6-15.5.1 Connection Paths to I/O Assembly Instances ..... 57
6-15.6 I/O Assembly Data Attribute Format ..... 58
6-15.7 Mapping I/O Assembly Data Attribute Components ..... 60
6-15.8 Defining Device Configuration ..... 61
6-15.8.1 Mapping Parameter Object Data ..... 62
6-15.8.2 Parameter Group Objects ..... 63
6-16 Servo Drive Device ..... 66
6-17 Barcode Scanner Device ..... 67
6-18 Position Controller Device ..... 68
6-18.1 Object Model ..... 68
6-18.1.1 Model Description ..... 68
6-18.2 How Objects Affect Behavior ..... 69
6-18.3 Defining Object Interfaces ..... 70
6-18.4 I/O Connection Messages ..... 70
6-18.4.1 Message Formats ..... 70
6-18.4.2 Definition of a Profile Move ..... 71
6-18.4.3 Starting a Profile Move ..... 71
6-18.4.4 I/O Handshaking Procedure ..... 72
6-18.5 I/O Connection Message Types ..... 76
6-18.5.1 Command Message Types ..... 76
6-18.5.2 Semantics ..... 79
6-18.5.2.1 Load Data/ Start Profile ..... 79
6-18.5.2.2 Start Block ..... 79
6-18.5.2.3 Incremental ..... 80
6-18.5.2.4 Direction (V. Mode) ..... 80
6-18.5.2.5 Smooth Stop ..... 80
6-18.5.2.6 Hard Stop ..... 80
6-18.5.2.7 Reg Arm ..... 80
6-18.5.2.8 Enable ..... 80


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

6-18.5.2.9 Block # ..... 80
6-18.5.2.10 Command Message Type ..... 80
6-18.5.2.11 Response Message Type ..... 80
6-18.5.2.12 Command Axis Number ..... 80
6-18.5.2.13 Target Position - Command Message Type 01 hex ..... 81
6-18.5.2.14 Target Velocity - Command Message Type 02 hex ..... 81
6-18.5.2.15 Acceleration - Command Message Type 03 hex ..... 81
6-18.5.2.16 Deceleration - Command Message Type 04 hex ..... 81
6-18.5.2.17 Torque - Command Message Type 05 hex ..... 81
6-18.5.2.18 Attribute Value – Command Message Types 19 – 1E hex ..... 81
6-18.5.2.19 Object Attribute to Get – Command Message Types 19 – 1D hex ..... 81
6-18.5.2.20 Object Attribute to Set – Command Message Types 19 – 1D hex ..... 81
6-18.5.2.21 Command Block Attribute to Get/Set – Command Message Type 1E hex ..... 81
6-18.5.2.22 Command Block Instance to Get/Set – Command Message Type 1E hex ..... 82
6-18.5.2.23 Parameter Instance to Get - Command Message Type 1F hex ..... 82
6-18.5.2.24 Parameter Instance to Set - Command Message Type 1F hex ..... 82
6-18.5.2.25 Parameter Value - Command Message Type 1F hex ..... 82
6-18.5.3 Response Message Types ..... 82
6-18.5.4 Semantics ..... 87
6-18.5.4.1 Profile in Progress ..... 87
6-18.5.4.2 Block in Execution ..... 87
6-18.5.4.3 On Target Position ..... 87
6-18.5.4.4 General Fault ..... 87
6-18.5.4.5 Current Direction ..... 87
6-18.5.4.6 Home Level ..... 87
6-18.5.4.7 Reg Level ..... 88
6-18.5.4.8 Enable ..... 88
6-18.5.4.9 Executing Block # ..... 88
6-18.5.4.10 Fault Input Fault ..... 88
6-18.5.4.11 Fwd Limit ..... 88
6-18.5.4.12 Rev Limit ..... 88
6-18.5.4.13 Positive Limit ..... 88
6-18.5.4.14 Negative Limit ..... 88
6-18.5.4.15 FE Fault ..... 88
6-18.5.4.16 Block Fault ..... 88
6-18.5.4.17 Load Complete ..... 88
6-18.5.4.18 Response Message Type ..... 88
6-18.5.4.19 Response Axis Number ..... 88
6-18.5.4.20 Actual Position - Response Message Type 01 hex ..... 89
6-18.5.4.21 Commanded Position - Response Message Type 02 hex ..... 89
6-18.5.4.22 Actual Velocity - Response Message Type 03 hex ..... 89
6-18.5.4.23 Command Velocity - Response Message Type 04 hex ..... 89
6-18.5.4.24 Torque - Response Message Type 05 hex ..... 89
6-18.5.4.25 Home Position - Response Message Type 06 hex ..... 89
6-18.5.4.26 Index Position - Response Message Type 07 hex ..... 89
6-18.5.4.27 Registration Position - Response Message Type 08 hex ..... 89
6-18.5.4.28 General Error Code – Response Message Type 14 hex ..... 89
6-18.5.4.29 Additional Code – Response Message Type 14 hex ..... 90
6-18.5.4.30 Attribute Value – Response Message Types 19 – 1E hex ..... 90
6-18.5.4.31 Object Attribute to Get – Response Message Types 19 – 1E hex ..... 90
6-18.5.4.32 Parameter Instance to Get - Response Message Type 1F hex ..... 90
6-18.5.4.33 Parameter Value - Response Message Type 1F hex ..... 90
6-18.6 Mapping I/O Message Data Attribute Components ..... 90
6-19 Motor Overload Device ..... 93
6-19.1 Object Model ..... 93
6-19.2 How Objects Affect Behavior ..... 94


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

6-19.3 Defining Object Interfaces ..... 95
6-19.4 Contactor Interface and Behavior ..... 95
6-19.5 I/O Assembly Instances ..... 96
6-19.5.1 Connection Paths to I/O Assembly Instances ..... 96
6-19.6 I/O Assembly Data Attribute Format ..... 97
6-19.6.1 Output Assembly Data Attribute Format ..... 97
6-19.6.2 Input Assembly Data Attribute Format ..... 97
6-19.7 Mapping I/O Assembly Data Attribute Components ..... 97
6-19.7.1 Mapping for Output Assembly Data Components ..... 97
6-19.7.2 Mapping for Input Assembly Data Components ..... 98
6-19.8 Defining Device Configuration ..... 98
6-20 Weigh Scale Device ..... 99
6-21 Encoder Device ..... 100
6-21.1 Introduction ..... 100
6-21.2 Object Model ..... 100
6-21.3 How Objects Affect Behavior ..... 101
6-21.4 Defining Object Interfaces ..... 101
6-21.5 I/O Assembly Instances ..... 101
6-21.6 I/O Assembly Data Attribute Format ..... 102
6-21.7 Mapping I/O Assembly Data Attribute Components ..... 103
6-21.8 Defining Device Configuration ..... 103
6-21.9 Mapping Parameter Object Data ..... 104
6-22 Resolver Device ..... 105
6-22.1 Object Model ..... 105
6-22.2 How Objects Affect Behavior ..... 106
6-22.3 Defining Object Interfaces ..... 106
6-22.4 I/O Assembly Instances ..... 106
6-22.5 I/O Assembly Data Attribute Format ..... 106
6-22.6 Mapping I/O Assembly Data Attribute Components ..... 107
6-22.7 Configuration Assembly Instances ..... 107
6-22.8 Configuration Assembly Data Attribute Format ..... 107
6-22.9 Mapping Configuration Assembly Data Attribute Components ..... 108
6-22.10 Defining Device Configuration ..... 108
6-22.10.1 Parameter Object Instances ..... 108
6-22.10.2 Mapping Parameter Object Data ..... 108
6-22.10.3 Configuration Parameter Definitions ..... 109
6-22.11 Effect of Configuration Parameters on Behavior ..... 110
6-23 Control Station Device ..... 111
6-24 Message Display Device ..... 112
6-25 Circuit Breaker ..... 113
6-26 Pneumatic Valve Device ..... 114
6-26.1 Object Model ..... 114
6-26.2 How Objects Affect Behavior ..... 115
6-26.3 Defining Object Interfaces ..... 115
6-26.4 I/O Assembly Instances ..... 115
6-26.5 I/O Assembly Data Attribute Format ..... 117
6-26.5.1 The I/O Assembly data attribute for *Solenoids Status Data* is shown below: ..... 117
6-26.5.2 The I/O Assembly data attribute for *General Purpose Discrete Input Data* is shown below: ..... 119
6-26.5.3 The I/O Assembly data attribute for *Solenoids Status Data* plus *Discrete Input Data* ..... 120
6-26.5.4 The I/O Assembly data attribute for the *Solenoid Valves Output Data* ..... 123
6-26.6 Mapping I/O Assembly Data Attribute Components ..... 125
6-26.7 Configuration Data and Public Interfaces ..... 126
6-26.8 Parameter Object Instances ..... 126
6-27 Contactor Device ..... 127
6-27.1 Object Model ..... 127
6-27.2 How Objects Affect Behavior ..... 128


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

6-27.3 Defining Object Interfaces ..... 128
6-27.4 Contactor Interface and Behavior ..... 129
6-27.5 I/O Assembly Instance ..... 130
6-27.5.1 Connection Paths to I/O Assembly Instances ..... 130
6-27.6 I/O Assembly Data Attribute Format ..... 131
6-27.7 Mapping I/O Assembly Data Attribute Components ..... 131
6-27.8 Defining Device Configuration ..... 131
6-28 Motor Starter Device ..... 132
6-28.1 Object Model ..... 132
6-28.2 How Objects Affect Behavior ..... 133
6-28.3 Defining Object Interfaces ..... 133
6-28.3.1 Starter Interface and Behavior ..... 134
6-28.3.2 Reversing Motor Starter Interface and Behavior ..... 135
6-28.3.3 Two Speed Motor Starter Interface and Behavior ..... 136
6-28.4 I/O Assembly Instances ..... 138
6-28.4.1 Connection Paths to I/O Assembly Instances ..... 138
6-28.5 I/O Assembly Data Attribute Format ..... 139
6-28.5.1 Output Assembly Data Attribute Format ..... 139
6-28.5.2 Input Assembly Data Attribute Format ..... 139
6-28.6 Mapping I/O Assembly Data Attribute Components ..... 140
6-28.6.1 Mapping for Motor Starter Output Assembly Data Components ..... 140
6-28.6.2 Mapping for Motor Starter Input Assembly Data Components ..... 140
6-28.7 Defining Device Configuration ..... 140
6-29 Softstart Starter Device ..... 141
6-29.1 Object Model ..... 141
6-29.2 How Objects Affect Behavior ..... 142
6-29.3 Defining Object Interfaces ..... 143
6-29.4 Softstart Motor Interface and Behavior ..... 144
6-29.5 I/O Assembly Instances ..... 146
6-29.5.1 Connection Paths to I/O Assembly Instances ..... 146
6-29.6 I/O Assembly Data Attribute Format ..... 147
6-29.6.1 Output Assembly Data Attribute Format ..... 147
6-29.6.2 Input Assembly Data Attribute Format ..... 147
6-29.7 Mapping I/O Assembly Data Attribute Components ..... 147
6-29.8 Defining Device Configuration ..... 147
6-30 Human-Machine Interface (HMI) ..... 148
6-30.1 Object Model ..... 148
6-30.2 How Objects Affect Behavior ..... 149
6-30.3 Defining Object Interfaces ..... 149
6-31 Mass Flow Controller Device ..... 150
6-31.1 Object Model ..... 150
6-31.1.1 Class Subclasses ..... 150
6-31.1.2 Instance Subclasses ..... 150
6-31.2 How Objects Affect Behavior ..... 152
6-31.3 Defining Object Interfaces ..... 152
6-31.4 I/O Assembly Instances ..... 153
6-31.5 I/O Assembly Object Instance Data Attribute Format ..... 153
6-31.6 Mapping I/O Assembly Data Attribute Components ..... 157
6-31.7 Object Limitations and Specific Definitions ..... 158
6-31.7.1 S-Device Supervisor Object Instance ..... 158
6-31.7.1.1 Limitations ..... 158
6-31.7.1.2 Specific Definition ..... 158
6-31.7.2 S-Analog Sensor Object Instance ..... 159
6-31.7.2.1 Limitations ..... 159
6-31.7.3 S-Analog Actuator Object Instance ..... 159
6-31.7.3.1 Limitations ..... 159


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

6-31.7.4 S-Single Stage Controller Object Instance ..... 160
6-31.7.4.1 Limitations ..... 160
6-31.8 Defining Device Configuration ..... 160
6-32 Vacuum/Pressure Gauge Device ..... 161
6-32.1 Object Model ..... 161
6-32.2 Class Subclasses ..... 162
6-32.3 Instance Subclasses ..... 162
6-32.4 How Objects Affect Behavior ..... 164
6-32.5 Defining Object Interfaces ..... 165
6-32.6 I/O Assembly Instances ..... 165
6-32.6.1 Mapping I/O Assembly Data Attribute Components ..... 171
6-32.7 Object Limitations and Specific Definitions ..... 172
6-32.7.1 S-Device Supervisor Object Instance ..... 172
6-32.7.1.1 Exception Attributes Definition ..... 172
6-32.7.1.2 Manufacturer's Device Type Attribute Definition ..... 174
6-32.7.1.3 Behavior ..... 174
6-32.7.2 S-Analog Sensor Object Instance ..... 175
6-32.7.2.1 Limitations ..... 175
6-32.7.2.2 Object Extensions ..... 175
6-32.7.3 Trip Point Object Instance ..... 176
6-32.7.3.1 Limitations ..... 176
6-32.8 Defining Device Configuration ..... 176
6-33 Programmable Logic Controller Device ..... 177
6-33.1 Object Model ..... 177
6-33.2 Defining Object Interfaces ..... 177
6-34 ControlNet Physical Layer Component Device ..... 178
6-35 Process Control Valve Device ..... 179
6-35.1 Object Model ..... 179
6-35.2 How Objects Affect Behavior ..... 181
6-35.3 Defining Object Interfaces ..... 182
6-35.3.1 I/O Assembly Instances ..... 182
6-35.3.2 Mapping I/O Assembly Data Attribute Components ..... 184
6-35.4 Object Limitations and Specific Definitions ..... 185
6-35.4.1 S-Device Supervisor Object ..... 185
6-35.4.1.1 Exception Attributes Definition ..... 185
6-35.4.2 S-Analog Sensor Object ..... 187
6-35.4.3 S-Single Stage Controller Object ..... 187
6-35.4.4 Selection Object ..... 188
6-35.4.5 Discrete Input Point Object ..... 189
6-35.5 Defining Device Configuration ..... 189
6-36 Turbomolecular Vacuum Pump Device ..... 190
6-36.1 Object Model ..... 190
6-36.2 Class Subclasses ..... 190
6-36.3 Instance Subclasses ..... 190
6-36.4 How Objects Affect Behavior ..... 192
6-36.5 Defining Object Interfaces ..... 192
6-36.6 I/O Assembly Instances ..... 192
6-36.7 I/O Assembly Instance Component Mapping ..... 194
6-36.8 Object Limitations and Specific Definitions ..... 195
6-36.8.1 S-Device Supervisor ..... 195
6-36.8.2 Register Object ..... 196
6-36.8.2.1 Object Instance Number Assignment ..... 196
6-36.8.3 Discrete Input Point Object ..... 197
6-36.8.3.1 Object Instance Number Assignment ..... 197
6-36.8.3.2 Attribute Definitions ..... 197
6-36.8.4 Discrete Output Point Object ..... 197


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

6-36.8.4.1 Object Instance Number Assignment ..... 197
6-36.8.5 AC/DC Drive Object ..... 198
6-36.8.5.1 Attribute Limitations ..... 198
6-36.8.5.2 Behavior ..... 199
6-36.8.6 S-Analog Sensor Object ..... 199
6-36.8.6.1 Object Instance Number Assignment ..... 199
6-36.8.6.2 Attribute Definitions ..... 200
6-36.8.7 S-Single Stage Controller Object ..... 200
6-36.8.7.1 Object Instance Number Assignment ..... 200
6-36.8.7.2 Attribute Definitions ..... 200
6-36.9 Defining Device Configuration ..... 200
6-37 Residual Gas Analyzer Device ..... 201
6-37.1 Object Model ..... 201
6-37.2 Class Subclasses ..... 201
6-37.3 Instance Subclasses ..... 201
6-37.4 How Objects Affect Behavior ..... 202
6-37.5 Defining Object Interfaces ..... 203
6-37.6 I/O Assembly Instances ..... 203
6-37.6.1 I/O Assembly Instance Component Formats ..... 205
6-37.6.2 I/O Assembly Instance Component Mapping ..... 205
6-37.7 Object Limitations and Specific Definitions ..... 206
6-37.7.1 S-Analog Sensor Object ..... 206
6-37.7.1.1 Class ..... 206
6-37.7.1.2 Instance ..... 206
6-37.7.1.3 Attribute Definitions ..... 207
6-37.7.2 Discrete Input Point Object Instance ..... 207
6-37.7.3 Discrete Output Point Object Instance ..... 207
6-37.7.3.1 Object Instance Number Assignment ..... 207
6-37.8 Defining Device Configuration ..... 207
6-38 DC Power Generator Device ..... 208
6-38.1 Object Model ..... 208
6-38.2 Subclasses ..... 208
6-38.3 How Objects Affect Behavior ..... 211
6-38.4 Defining Object Interfaces ..... 211
6-38.5 I/O Assembly Instances ..... 211
6-38.5.1 I/O Assembly Instance Component Formats ..... 214
6-38.5.2 I/O Assembly Instance Component Mapping ..... 214
6-38.6 Object Limitations and Specific Definitions ..... 216
6-38.6.1 S-Device Supervisor ..... 216
6-38.6.2 S-Analog Sensor Object Instances ..... 217
6-38.6.3 S-Analog Actuator Object Instances ..... 217
6-38.6.4 S-Single Stage Controller Object Instances ..... 218
6-38.6.4.1 Behavior ..... 218
6-38.6.4.2 Selection Object, Instance 1 ..... 218
6-38.6.5 Register Object ..... 219
6-38.6.5.1 Register Object, Instance 1 (Input) Operational Status ..... 219
6-38.6.5.2 Register Object, Instance 2 (Input) Interlock Status ..... 219
6-38.6.5.3 Behavior ..... 219
6-38.7 Defining Device Configuration ..... 219
6-39 RF Power Generator Device ..... 220
6-39.1 Object Model ..... 220
6-39.2 Subclasses ..... 220
6-39.3 How Objects Affect Behavior ..... 223
6-39.4 Defining Object Interfaces ..... 223
6-39.5 I/O Assembly Instances ..... 223
6-39.5.1 I/O Assembly Instance Component Formats ..... 226


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

6-39.5.2 I/O Assembly Instance Component Mapping ..... 226
6-39.6 Object Limitations and Specific Definitions ..... 227
6-39.6.1 S-Device Supervisor ..... 228
6-39.6.2 S-Analog Sensor Object Instances ..... 229
6-39.6.3 S-Analog Actuator Object Instances ..... 230
6-39.6.3.1 S-Single Stage Controller Object Instances ..... 230
6-39.6.3.2 Behavior ..... 230
6-39.6.4 Selection Object, Instance 1 ..... 231
6-39.6.5 Register Object ..... 231
6-39.6.5.1 Register Object, Instance 1 Operational Status ..... 231
6-39.6.5.2 Register Object, Instance 2 Interlock Status ..... 232
6-39.6.5.3 Behavior ..... 232
6-39.7 Defining Device Configuration ..... 232
6-40 Fluid Flow Controller Device ..... 233
6-40.1 Object Model ..... 233
6-40.2 Class Subclasses ..... 233
6-40.3 Instance Subclasses ..... 234
6-40.4 Instance Identifiers ..... 234
6-40.5 How Objects Affect Behavior ..... 236
6-40.6 Defining Object Interfaces ..... 236
6-40.7 I/O Assembly Instances ..... 236
6-40.8 I/O Assembly Object Instance Data Attribute Encoding ..... 237
6-40.9 Mapping I/O Assembly Data Attribute Components ..... 238
6-40.10 Object Limitations and Specific Definitions ..... 238
6-40.10.1 S-Device Supervisor Object Instance ..... 238
6-40.10.2 Specific Definition ..... 238
6-40.10.3 S-Analog Sensor Object ..... 239
6-40.10.4 S-Analog Actuator Object ..... 240
6-40.10.5 S-Single Stage Controller Object ..... 240
6-40.11 Defining Device Configuration ..... 241
6-41 Enhanced Mass Flow Controller Device ..... 242
6-41.1 Introduction ..... 242
6-41.2 Object Model ..... 242
6-41.3 Class Subclasses ..... 243
6-41.4 Instance Subclasses ..... 243
6-41.5 How Objects Affect Behavior ..... 244
6-41.6 Defining Object Interfaces ..... 245
6-41.7 I/O Assembly Instances ..... 245
6-41.8 I/O Assembly Object Instance Data Attribute Format ..... 246
6-41.8.1 Mapping I/O Assembly Data Attribute Components ..... 250
6-41.9 Object Limitations and Specific Definitions ..... 251
6-41.9.1 S-Device Supervisor Object Instance ..... 251
6-41.9.2 S-Analog Sensor Object ..... 252
6-41.9.3 S-Analog Sensor Object ..... 253
6-41.9.4 S-Analog Actuator Object ..... 253
6-41.9.5 S-Single Stage Controller Object ..... 253
6-41.10 Defining Device Configuration ..... 253
6-42 CIP-Modbus Translator Device ..... 254
6-42.1 Object Model ..... 254
6-42.2 How Objects Affect Behavior ..... 255
6-42.3 Defining Object Interfaces ..... 255
6-43 Embedded Component ..... 256
6-43.1 Identity Object Instance Attribute Requirements ..... 256
6-43.1.1 Product Code (Attribute 3) ..... 257
6-43.1.2 Revision (Attribute 4) ..... 257
6-43.2 Identity Object Instance Service Requirements ..... 257


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-1 Introduction

To provide interoperability and promote interchangeability by like device types, there must be some consistency between devices of the same type. That is, there must be a core "standard" for each device type. In general, like devices must:

- exhibit the same behavior
- produce and/or consume the same basic set of I/O data
- contain the same basic set of configurable attributes

The formal definition of this information is known as a ***device profile***. This chapter provides a detailed definition of a device profile and describes its components.

A device profile **shall** contain:

- an object model for the device type
- the I/O data format for the device type configuration data and the public interface(s) to that data

You may adopt or extend one of the existing profiles in this chapter or you may define your own profile based on the format contained in this chapter.

## 6-2 The Object Model

To provide interoperability among like devices, the same object implemented in two or more devices **shall** behave identically from device to device. Consequently, each object specification includes a rigid definition of behavior.

Every CIP product contains several objects. These objects interact to provide basic product behavior. Because the behavior of individual objects is fixed, the behavior of identical groupings of objects is also fixed. Therefore, the same group of objects arranged in a specified order will interact to produce the same behavior from device to device.

The *grouping* of objects used in a device is referred to as that device's ***object model***. See Figure 6-2.1. For like devices to produce identical behavior, they must have identical object models. Therefore, an object model is included with every device profile to provide for interoperability among like CIP devices.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-2.1Object Model**

An object model specification:

- Identifies all object classes present in the device (required, optional and conditional).
- Indicates the number of instances present in each object class. If the device supports the dynamic creation and deletion of instances, then the object model states the maximum number of instances that can exist within the object class.
- States whether or not the object affects behavior of the device. If it does affect behavior, the object model states how.
- Defines the interface to each object. This defines how objects and object classes are linked.

## 6-2.1 All Objects Present in a Device

Every device can contain both required objects, optional, and conditional objects. When an object is identified as "required," it is required for ***all*** devices of that type. Each device profile shall contain an Object Interface Table, which shall list the interface to each object. At a minimum, every object model for a CIP device must specify instances of these object classes:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-2.1 Minimum Objects Required For All Devices**

| Object Class | | Optional/Required | # of Instances |
|---|---|---|---|
| | Identity Object | Required | 1 |
| | Message Router Object | Required | 1 |
| Network Specific Link Object(s) | DeviceNet Object | Conditional[1] | 1 |
| | Ethernet Link Object | Conditional[2] | 1 |
| | TCP/IP Interface Object | Conditional[2] | 1 |
| | ControlNet Object | Conditional[3] | 1 |
| | CompoNet Link Object | Conditional[4] | 1 |
| | Connection Object | Conditional[1, 5] | 2 (explicit, I/O) [5] |
| | Connection Manager Object | Conditional[2 - 3] | 1 |

Table Footnotes
1 Required if CIP network type is DeviceNet
2 Required if CIP network type is EtherNet/IP
3 Required if CIP network type is ControlNet
4 Required if CIP network type is CompoNet
5 If CIP network type is CompoNet, see Chapter 6 of Volume 6, CompoNet Adaptation of CIP

Although not an object, each device shall also support the Unconnected Message Manager (UCMM). In addition to these minimum object classes, object models can, and probably will, contain application–specific object classes that are required by the device type.

Some object classes may be included that provide functions beyond the minimum required of a particular device type or that have no effect on device behavior. These types of objects are identified in the profile as “optional” or “conditional.” When an object is identified as “optional,” it is optional for ***all*** devices of that type. The device type dictates what objects are necessary to provide the device’s required basic function. When an object is identified as “conditional,” it is required “if” a specified condition exists. The Device Profile shall specify the conditions.

**Important:** Instances of OPTIONAL object classes may provide behavior beyond the behavior defined for the device type. At power up, however, this additional behavior shall default in a manner such that the device’s behavior appears to be identical to the basic behavior defined for that device type.

**Table 6-2.2 Required/Optional Object Criteria**

| Object classes are specified as REQUIRED when they | Object classes are specified as OPTIONAL when they |
|---|---|
| affect in any way the basic behavior specified for the device type | provide behavior beyond the minimum specified for the device type |
| are used to define the I/O data format of the device | provide functions beyond the minimum required of a particular device type or have NO effect on behavior of the device |
| provide the primary method of access to the device’s configuration data | provide an optional method of access to the device’s configuration data. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-2.2 Objects That Affect Behavior

After all objects included in the device are identified, this section of a profile distinguishes between objects that do and do not affect the behavior of the device. If an object affects behavior, this section states how. Any component (object, attribute, or service) that affects the behavior of a device is specified here. The following table shows the format of this part of the device profile description.

**Table 6-2.3 Components That Affect Behavior of the Device**

| Component | Effect on behavior |
|---|---|
| Attribute/Object | Behavior |

### 6-2.3 Object Interfaces

The final portion of the object model specification within a device profile is the definition of all interfaces to each of the device's internal objects. Defining object interfaces indicates how the objects within a device are connected.

**Table 6-2.4 Object Interfaces**

| Object | Interface |
|---|---|
| Name of Object | Name of Interface (Explicit Messaging Connection Instance, Message Router, I/O connection) |

In summary, an object model defines behavior of a device in the following terms:

- objects present in device
- maximum number of object instances
- how objects affect behavior
- object interfaces

## 6-3 I/O Data Format

This section of a profile defines how a device communicates on the CIP network, which includes an exact specification of the device's I/O data format.

Smart networked devices can (and probably will) produce and/or consume more than one I/O value. Typically, they will produce and/or consume one or more I/O values, as well as status and diagnostic information. Each piece of data communicated by a device is represented by an attribute of one of the device's internal objects.

Communicating multiple pieces of data (attributes) across a single I/O connection requires that the attributes be grouped or assembled together into a single data block. Instances of the ***Assembly Object Class*** perform this grouping. Thus, the definition of a device's I/O data format is equivalent to the definition of the assembly instances used to group the device's I/O data.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

In a device profile, the I/O data format of devices adheres to these guidelines:

- I/O Assemblies are either Input type or Output type
- A device may contain more than one I/O assembly (data format of I/O instances may be a configurable option of your device)

The definition of a device's I/O assembly instances:

- Identifies the I/O assembly by instance number, type, and name
- Specifies the I/O assembly Data attribute format
- Maps the I/O assembly Data attribute components to other attributes

### 6-3.1 I/O Assembly Instances

Because CIP products can contain one or more I/O assemblies (of either Input type or Output type), assembly instances are clearly identified for each device type. The following table identifies the I/O assembly instance supported by a typical device.

**Table 6-3.1 Identifying I/O Assembly Instances**

| Number | Type | Name |
|---|---|---|
| Assembly # | Input or Output | Name of assembly |

### 6-3.2 Format of I/O Assembly Data Attribute

Any device communicating I/O data to and from another device must have knowledge of the other device's I/O data format. The *Data* attribute of the Assembly Object (instance attribute #3) holds this I/O format. Therefore, this section of a profile specifies the format of the Data attribute for **each** assembly instance listed in the assembly instance identification table.

The Data attribute is an array of bytes. The device profile specifies how that array is defined to represent a device's I/O data.

Specification of the I/O assembly Data attribute format adheres to these guidelines:

- List Data components that are larger than one byte in size with the low–order byte first
- Right justify within a byte (starting with bit 0) Data components that are smaller than one byte
- Explicitly state if bits or bytes are to be reserved

### 6-3.3 Map of I/O Assembly Data Attribute Components

Because components of the Assembly Object's *Data* attribute are attributes of other objects, a device profile contains a mapping of those attributes to their respective objects.

The map includes specification of the member path (Class ID, Instance ID, etc.) for each data component. Specification of the relative addresses of each Data attribute component is essentially equivalent to specification of the ***Member_List*** instance attribute (#2) of the Assembly Object.

The following table shows the format for an I/O assembly Data attribute mapping.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-3.2 Example I/O Assembly Data Attribute Mapping**

| Data Component | Class | | Instance | Attribute | | Data Type |
|---|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number | |
| Component name within profile | Component class name within object library | $xx_{hex}$ | Y | Component attribute name within object library | z | |

If a device has more than one I/O assembly instance, the profile should include a table similar to the one above for **each** I/O assembly instance.

## 6-4 Device Configuration

In addition to a product's object model and format of its I/O data, a device profile includes specification of the device's configurable parameters and the public interface to those parameters.

The configurable parameters in a device directly affect its behavior. Because like devices must behave in an identical fashion, they **must** have identical configuration parameters.

**Important:** "Identical configuration" refers to *basic* configuration. A device may have extended functionality (with associated parameters) that is beyond the behavior defined for the device type. At power up, this functionality must default in a manner such that the device's behavior appears to be identical to the behavior defined for that device type.

In addition to defining identical configuration parameters, the public interfaces to those parameters **must** be identical.

Definition of a device's configuration includes the following information for ***each*** configurable attribute:

- configuration parameter data:
  - all attribute values of each Parameter Object Instance
  - all values in the parameter section of an Electronic Data Sheet
  - at minimum, the following printed data sheet information:
    - parameter name
    - attribute path (class, instance, attribute)
    - data type
    - parameter units
    - minimum/maximum default values
- effect of parameters on device behavior
- parameter groups if any configurable parameters are grouped using an instance of the Parameter Group Object Class
- public interface to the device's configuration (i.e., bulk configuration via a configuration assembly, full/stub instances of the Parameter Object Class, etc.)


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-4.1 Parameter Data

The definition of each configuration parameter includes specification of one of the following:

- the instance attributes of an instance of the Parameter Object Class (for each of your configuration parameters)
- all data outlined in the parameter section of an EDS
- at minimum, the following printed data sheet information:
  - parameter name
  - attribute path (class, instance, attribute)
  - data type
  - parameter units
  - minimum/maximum default values

### 6-4.2 Effect of Configuration Parameters on Behavior

The effect that each of the configuration parameters has on the device's behavior is also documented in the configuration section of a device profile. The following table shall be used within the device profile.

**Table 6-4.1 Example of Table Showing the Effect of Parameters on Behavior**

| Parameter | Effect on Behavior |
|---|---|
| Parameter name | Effect |

### 6-4.3 Parameter Groups

If any configurable parameters are grouped using an instance of the *Parameter Group Object Class*, then the definition of each group is specified in this section.

The definition of each configuration parameter group includes specification of either:

- the instance attributes of an instance of the Parameter Group Object Class (for each of your configuration parameters); or
- all data outlined in the parameter group section of an EDS

### 6-4.4 Public Interfaces to Device Configuration Data

The final portion of the configuration section of a profile clearly specifies the public interface(s) to a device's configuration data.

#### 6-4.4.1 Parameter Object

If a device employs instances of the Parameter Object Class, each instance and the configuration parameter associated with it is specified here. Also included here is a map of the configuration parameter to the object in which it is contained.

**Table 6-4.2 Parameter Instance Listing**

| Instance Number | Configuration Parameter Name |
|---|---|
| X | Parameter name |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-4.3 Configuration Parameter Mapping**

| Configuration Parameter Name | Class | | Instance Number | Attribute | | Data Type |
|---|---|---|---|---|---|---|
| | Name | Number | | Name | Number | |
| Parameter name | Class | $xx_{hex}$ | y | Attribute | z | |

### 6-4.4.2 Configuration Assembly Object

Documentation of a device's configuration assembly provides information similar to that which is specified for the device's I/O assemblies. This section of a profile includes:

- specification of the configuration assembly Data attribute format
- mapping of each configurable attribute using its logical address (Class/Instance/Attribute)

Specification of the configuration assembly Data attribute format adheres to these guidelines:

- List Data components that are larger than one byte in size with the low–order byte first
- Right justify within a byte (starting with bit 0) Data components that are smaller than one byte
- Explicitly state if bits or bytes are to be reserved

The table below shows how the format of the Configuration Assembly Object's Data attribute is specified.

**Table 6-4.4 Configuration Assembly Data Attribute Format**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Configuration Parameter 1 | | | | | | | |
| 1 | Configuration Parameter 2 | | | | | | | |
| 2 | Configuration Parameter 3 | | | | | | | |

In addition to specification of the device's configuration assembly Data attribute format, this section also includes a mapping of the individual configuration assembly Data attribute components to their respective objects.

The map includes specification of the Class, Instance, and Attribute IDs for each data component. Specification of the relative addresses of each Data attribute component is essentially equivalent to specification of the ***Member_List*** instance attribute (#2) of the Assembly Object. The table below shows the format for the configuration assembly Data attribute mapping.

**Table 6-4.5 Configuration Assembly Data Attribute Mapping**

| Configuration Parameter Name | Class | | Instance Number | Attribute | | Data Type |
|---|---|---|---|---|---|---|
| | Name | Number | | Name | Number | |
| Configuration Parameter1 | Class | $xx_{hex}$ | y | Range | Z | |

## 6-5 Extended Device Profiles

You have the option of adopting existing device profiles and then extending them to incorporate any additional behavior your product may exhibit.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Manufacturers of multiple source products may wish to design a product such that it provides the basic behavior defined in the product's device profile and, in addition, provides extended functionality that helps distinguish one product from another.

**Important:** The basic device profile definition must not change when extending an existing device profile. Also, the added functionality must not make the extended profile incompatible with the basic device profile. For these reasons, you must adhere to the following rules when extending an existing device profile:

- All new objects, attributes, and services added to the profile are OPTIONAL. Backwards compatibility shall be maintained.
- At power–up, all new behavior must default such that the device's behavior appears identical to the specified default behavior defined for the device type.
- The basic I/O format must not change. Extended I/O formats can be provided for by adding optional I/O assembly instances.
- The basic configuration must not change. Extended configuration parameters can be provided for by adding optional configuration assembly instances or optional instances of the Parameter Object Class.
- Any additional assembly instances must be defined in the vendor–specific address range.

**Important:** Instances of the Assembly Class are divided into address ranges to provide for extensions to device profiles. See the Assembly Object definition in the object library.

## 6-6 Device Profile Numbering Scheme

The table below reveals the numbering scheme to be used for device profile numbering. The table shows that a device profile may be either publicly defined or vendor specific:

**Table 6-6.1 Device Profile Numbering**

| Type | Range | Quantity |
|---|---|---|
| Publicly Defined | $00_{hex}$ - $63_{hex}$ | 100 |
| Vendor Specific | $64_{hex}$ - $C7_{hex}$ | 100 |
| Publicly Defined | $C8_{hex}$ | 1 |
| Reserved by CIP | $C9_{hex}$ - $FF_{hex}$ | 55 |
| Publicly Defined | $100_{hex}$ - $2FF_{hex}$ | 512 |
| Vendor Specific | $300_{hex}$ - $4FF_{hex}$ | 512 |
| Reserved by CIP | $500_{hex}$ - $FFFF_{hex}$ | 64,256 |

While you are highly encouraged to adopt or develop a device profile for your product you may be unwilling or unable to do so. For this reason ranges of device type numbers have been set aside for "Vendor Specific" device profiles. If you choose to use one of these device type numbers you are not required to publish a device profile for your product. It is important to note, however, that if you do not publish your device's profile, your customers will not be able to find direct replacements for your product and, more importantly, they will not be able to use your product as a direct replacement for your competitor's product. Additionally, even vendor specific device profiles are required to support the minimum objects listed in section 6-2.1.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-7 Device Profiles

The remainder of this chapter contains listings of all existing device profiles at the time of publication.

**Table 6-7.1 Device Profiles Sorted by Name**

| Device Profile Name | Device Type Number | Section | Page |
|---|---|---|---|
| AC Drive Device | 0x02 | 6-15 | 6-53 |
| Barcode Scanner | Not yet assigned | 6-17 | 6-67 |
| CIP Modbus Device | 0x28 | See Volume 7, Integration of Modbus Devices in the CIP Architecture | |
| CIP Modbus Translator | 0x29 | 6-42 | 6-254 |
| CIP Motion Converter | 0x30 | See Volume 9, CIP Motion | |
| CIP Motion Drive | 0x25 | See Volume 9, CIP Motion | |
| CIP Motion Encoder | 0x2F | See Volume 9, CIP Motion | |
| CIP Motion I/O | 0x31 | See Volume 9, CIP Motion | |
| CIP Motion Safety Drive Device | 0x2D | See Volume 5, CIP Safety | |
| Circuit Breaker | Not yet assigned | 6-25 | 6-113 |
| Communications Adapter | 0x0C | 6-13 | 6-50 |
| CompoNet Repeater | 0x26 | See Volume 6, CompoNet Adaptation of CIP | |
| Contactor | 0x15 | 6-27 | 6-127 |
| Control Station | Not yet assigned | 6-23 | 6-111 |
| ControlNet Physical Layer Component | 0x32 | 6-34 | 6-178 |
| DC Drive | 0x13 | 6-15 | 6-53 |
| DC Power Generator | 0x1F | 6-38 | 6-208 |
| Embedded Component | 0xC8 | 6-43 | 6-256 |
| Encoder | 0x22 | 6-21 | 6-100 |
| Fluid Flow Controller | 0x24 | 6-40 | 6-233 |
| General Purpose Analog I/O | Not yet assigned | 6-14 | 6-52 |
| General Purpose Discrete I/O | 0x07 | 6-12 | 6-36 |
| Generic Device (deprecated for new devices) | 0x00 | 6-8 | 6-22 |
| Generic Device (keyable) | 0x2B | 6-8 | 6-23 |
| Human-Machine Interface | 0x18 | 6-30 | 6-148 |
| Inductive Proximity Switch | 0x05 | 6-10 | 6-28 |
| Limit Switch | 0x04 | 6-24 | 6-9 |
| Managed Ethernet Switch | 0x2C | See Volume 2, EtherNet/IP Adaptation of CIP | |
| Mass Flow Controller | 0x1A | 6-31 | 6-150 |
| Mass Flow Controller, Enhanced | 0x27 | 6-41 | 6-242 |
| Message Display | Not yet assigned | 6-24 | 6-112 |
| Motor Overload | 0x03 | 6-19 | 6-93 |
| Motor Starter | 0x16 | 6-28 | 6-132 |
| Photoelectric Sensor | 0x06 | 6-11 | 6-32 |
| Pneumatic Valve(s) | 0x1B | 6-26 | 6-114 |
| Position Controller | 0x10 | 6-18 | 6-68 |
| Process Control Valve | 0x1D | 6-35 | 6-179 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Device Profile Name | Device Type Number | Section | Page |
|---|---|---|---|
| Programmable Logic Controller | 0x0E | 6-33 | 6-177 |
| Residual Gas Analyzer | 0x1E | 6-37 | 6-201 |
| Resolver | 0x09 | 6-22 | 6-105 |
| RF Power Generator | 0x20 | 6-39 | 6-220 |
| Safety Analog I/O Device | 0x2A | See Volume 5, CIP Safety | |
| Safety Discrete I/O Device | 0x23 | See Volume 5, CIP Safety | |
| Safety Drive Device | 0x2E | See Volume 5, CIP Safety | |
| Servo Drives | Not yet assigned | 6-16 | 6-66 |
| Softstart Starter | 0x17 | 6-29 | 6-141 |
| Turbomolecular Vacuum Pump | 0x21 | 6-36 | 6-190 |
| Vacuum Pressure Gauge | 0x1C | 6-32 | 6-161 |
| Weigh Scale | Not yet assigned | 6-20 | 6-99 |

**Table 6-7.2 Device Profiles Sorted by Device Type Number**

| Device Type Number | Device Profile Name | Section | Page |
|---|---|---|---|
| 0x00 | Generic Device (deprecated for new devices) | 6-8 | 6-22 |
| 0x02 | AC Drive Device | 6-15 | 6-53 |
| 0x03 | Motor Overload | 6-19 | 6-93 |
| 0x04 | Limit Switch | 6-9 | 6-24 |
| 0x05 | Inductive Proximity Switch | 6-10 | 6-28 |
| 0x06 | Photoelectric Sensor | 6-11 | 6-32 |
| 0x07 | General Purpose Discrete I/O | 6-12 | 6-36 |
| 0x09 | Resolver | 6-22 | 6-105 |
| 0x0C | Communications Adapter | 6-13 | 6-50 |
| 0x0E | Programmable Logic Controller | 6-33 | 6-177 |
| 0x10 | Position Controller | 6-18 | 6-68 |
| 0x13 | DC Drive | 6-15 | 6-53 |
| 0x15 | Contactor | 6-27 | 6-127 |
| 0x16 | Motor Starter | 6-28 | 6-132 |
| 0x17 | Softstart Starter | 6-29 | 6-141 |
| 0x18 | Human-Machine Interface | 6-30 | 6-148 |
| 0x1A | Mass Flow Controller | 6-31 | 6-150 |
| 0x1B | Pneumatic Valve(s) | 6-26 | 6-114 |
| 0x1C | Vacuum Pressure Gauge | 6-32 | 6-161 |
| 0x1D | Process Control Valve | 6-35 | 6-179 |
| 0x1E | Residual Gas Analyzer | 6-37 | 6-201 |
| 0x1F | DC Power Generator | 6-38 | 6-208 |
| 0x20 | RF Power Generator | 6-39 | 6-220 |
| 0x21 | Turbomolecular Vacuum Pump | 6-36 | 6-190 |
| 0x22 | Encoder | 6-21 | 6-100 |
| 0x23 | Safety Discrete I/O Device | See Volume 5, CIP Safety | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Device Type Number | Device Profile Name | Section | Page |
|---|---|---|---|
| 0x24 | Fluid Flow Controller | 6-40 | 6-233 |
| 0x25 | CIP Motion Drive | See Volume 9, CIP Motion | |
| 0x26 | CompoNet Repeater | See Volume 6, CompoNet Adaptation of CIP | |
| 0x27 | Mass Flow Controller, Enhanced | 6-41 | 6-242 |
| 0x28 | CIP Modbus Device | See Volume 7, Integration of Modbus Devices in the CIP Architecture | |
| 0x29 | CIP Modbus Translator | 6-42 | 6-254 |
| 0x2A | Safety Analog I/O Device | See Volume 5, CIP Safety | |
| 0x2B | Generic Device (keyable) | 6-8 | 6-23 |
| 0x2C | Managed Ethernet Switch | See Volume 2, EtherNet/IP Adaptation of CIP | |
| 0x2D | CIP Motion Safety Drive Device | See Volume 5, CIP Safety | |
| 0x2E | Safety Drive Device | See Volume 5, CIP Safety | |
| 0x2F | CIP Motion Encoder | See Volume 9, CIP Motion | |
| 0x30 | CIP Motion Converter | See Volume 9, CIP Motion | |
| 0x31 | CIP Motion I/O | See Volume 9, CIP Motion | |
| 0x32 | ControlNet Physical Layer Component | 6-34 | 6-178 |
| 0xC8 | Embedded Component | 6-43 | 6-256 |
| Not yet assigned | Barcode Scanner | 6-17 | 6-67 |
| Not yet assigned | Circuit Breaker | 6-25 | 6-113 |
| Not yet assigned | Control Station | 6-23 | 6-111 |
| Not yet assigned | General Purpose Analog I/O | 6-14 | 6-52 |
| Not yet assigned | Message Display | 6-24 | 6-112 |
| Not yet assigned | Servo Drives | 6-16 | 6-66 |
| Not yet assigned | Weigh Scale | 6-20 | 6-99 |

The following device type numbers have been obsoleted

**Table 6-7.3 Obsolete Device Profiles**

| Obsoleted Device Type Number | Previous Profile Assignment |
|---|---|
| $01_{hex}$ | Control Station |
| $08_{hex}$ | Encoder |
| $0A_{hex}$ | General Purpose Analog I/O |
| $0D_{hex}$ | Barcode Scanner |
| $11_{hex}$ | Weigh Scale |
| $12_{hex}$ | Message Display |
| $14_{hex}$ | Servo Drives |
| $19_{hex}$ | Pneumatic Valve(s) |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-8 Generic Device

## Device Type: 2B Hex

NOTE: The device type ID of 0 that was previously assigned to this profile has been deprecated because 0 does not allow for exact matching when an Electronic Key segment is used. New devices shall not use device type ID 0.

The Generic Device type defines a device that does not fit into any of the defined device types. Initially, there will probably be many Generic Device type devices, but over time, ODVA Special Interest Groups (SIGs) will create a specific device profile for devices with similar functionality. The Generic Device type devices are not interchangeable.

## 6-8.1 Object Model

The Object Model in Table 6-8.1 represents the minimum support in a Generic Device. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-8.1 Objects Present in a Generic Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Required | at least 1 |
| Application | Required | at least 1 |

The Generic Device profile does not specify any instances of the Assembly Object or the type of application objects necessary for device operation. This portion of the device profile must be supplied by the product developer as described in Sections 6-1 through 6-7 inclusive.

Since the Generic Device profile does not define any static Assembly object instances, use of Assembly object instances in the "Open - Defined by the device profile" ranges (see Table 5A-5.2 Assembly Instance ID Ranges) for vendor specific purposes is permitted.

**Figure 6-8.1Object Model for the Generic Device**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-8.2 How Objects Affect Behavior

The objects for this device affect the device's behavior as shown in the table below.

**Table 6-8.2 Object Effect on Behavior**

| Object | Effect on behavior |
|---|---|
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines input/output and configuration data format |
| Application | Defines device operation |

## 6-8.3 Defining Object Interfaces

The objects in the Generic Device have the interfaces listed in the following table:

**Table 6-8.3 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| Application | Assembly or Message Router |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-9 Limit Switch Device

## Device Type: 04 Hex

A limit switch mechanically detects the presence or absence of a physical target object. The switch detects an object when a lever or rod makes physical contact with the object.

## 6-9.1 Object Model

The Object Model in Figure 6-9.1 represents a limit switch. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

The CIP Object Library provides more details about these objects.

**Table 6-9.1 Objects Present in a Limit Switch**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Required | 1 |
| Parameter | Optional | 1 |
| Presence Sensing | Required | 1 |

**Figure 6-9.1 Object Model for a Limit Switch**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-9.2 How Objects Affect Behavior

The objects for this device affect the device's behavior as shown in the table below.

**Table 6-9.2 Object Effect on Behavior**

| Object | Effect on behavior |
|---|---|
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines I/O data format |
| Parameter | Provides a public interface to the device's configuration data |
| Presence Sensing | Affects *Output Value* (attribute) |

## 6-9.3 Defining Object Interfaces

The objects in this device have the interfaces listed in the following table:

**Table 6-9.3 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| Parameter | Message Router |
| Presence Sensing | Message Router, Assembly Object, or Parameter Object |

## 6-9.4 I/O Assembly Instances

The following table identifies the I/O assembly instance supported by the limit switch.

**Table 6-9.4 I/O Assembly Instances**

| Number | Type | Name |
|---|---|---|
| 1 | Input | Input Data |

## 6-9.5 I/O Assembly Data Attribute Format

The I/O Assembly data attribute has the format shown below.

**Table 6-9.5 I/O Assembly Data Attribute Format**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Diagnostic | Output |

## 6-9.6 Mapping I/O Assembly Data Attribute Components

The following table indicates the I/O assembly Data attribute mapping for this limit switch device.

**Table 6-9.6 I/O Assembly Data Mapping**

| Data Component | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Diagnostic | presence sensing | $0E_{hex}$ | 1 | Diagnostic | 4 |
| Output | presence sensing | $0E_{hex}$ | 1 | Output | 1 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-9.7 Defining Device Configuration

Public access to the Presence Sensing Object by the Message Router must be supported for configuration of this device type. If supported, the optional Parameter Object may be used to access the device type's configuration parameter.

### 6-9.7.1 Parameter Object Instances

The limit switch contains one instance of the Parameter Object Class. This instance is a Parameter Object stub. See The CIP Object Library for the definition of the Parameter Object and an explanation of how it is used for configuration.

The following table identifies the Parameter Object instance supported by the limit switch.

**Table 6-9.7 Parameter Object Instances Supported**

| Number | Name |
|---|---|
| 1 | Operation Mode Configuration |

### 6-9.7.2 Mapping Parameter Object Data

The following table indicates the Parameter Object data mapping for the limit switch device.

**Table 6-9.8 Parameter Object Data Mapping**

| Configuration Parameter | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Operate Mode Configuration | presence sensing | 0Ehex | 1 | Operate Mode | 8 |

### 6-9.7.3 Configuration Parameter Definitions

The following sections of an example EDS show the information necessary to define the configuration parameters for a limit switch.

```
[Params]
      Param1=                 $Operate Mode
      0,                      $Data Placeholder
      2,”20 0e 24 01 30 08”,  $Path size and Path to Operate Mode Attr
      0x0002,                 $Descriptor (support enumerated strings
      0xC1,1                  $Data Type and Size (Boolean)
      ”Operate Mode”,         $Name
      ””,                     $Units (not used)
      ””,                     $User Manual Ref (not used)
      0,1,0,                  $min, max, default values
      0,0,0,0,                $mult, div, base, offset scaling (not used)
      0,0,0,0,                $mult, div, base, offset links (not used)
      1;                      $decimal places
[EnumPar]
      Param1=                 $Operate Mode Enumerated Strings
      ”Normally Open”,        $For value=0
      ”Normally Closed”;      $For value=1
```

## 6-9.8 Effect of Configuration Parameters on Behavior

The configuration parameter affects the device's behavior as shown below.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-9.9 Configuration Parameter Effect on Behavior**

| Parameter | Effect on Behavior |
|---|---|
| Operate Mode | Inverts the level defined for the Output attribute of the Presence Sensing Object |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-10 Inductive Proximity Switch Device

**Device Type: 05 Hex**

An inductive proximity switch operates in an electromagnetic field. When it senses a change in the field, it sends a signal to an output amplifier circuit to change the state of the circuit.

## 6-10.1 Object Model

The Object Model in Figure 6-10.1 represents an inductive proximity switch. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

The CIP Object Library provides more details about these objects.

**Table 6-10.1 Objects Present in an Inductive Proximity Switch**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Required | 1 |
| Parameter | Optional | 1 |
| Presence Sensing | Required | 1 |

**Figure 6-10.1 Object Model for an Inductive Proximity Switch**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-10.2 How Objects Affect Behavior

The objects for this device affect the device's behavior as shown in the table below.

**Table 6-10.2 Object Effect on Behavior**

| Object | Effect on behavior |
|---|---|
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines I/O data format |
| Parameter | Provides a public interface to the device's configuration data |
| Presence Sensing | Effects *Output Value* (attribute) |

## 6-10.3 Defining Object Interfaces

The objects in this device have the interfaces listed in the following table:

**Table 6-10.3 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| Parameter | Message Router |
| Presence Sensing | Message Router, Assembly Object, or Parameter Object |

## 6-10.4 I/O Assembly Instances

The following table identifies the I/O assembly instance supported by the inductive proximity switch.

**Table 6-10.4 I/O Assembly Instances**

| Number | Type | Name |
|---|---|---|
| 1 | Input | Input Data |

## 6-10.5 I/O Assembly Data Attribute Format

The I/O Assembly data attribute has the format shown below.

**Table 6-10.5 I/O Assembly Data Attribute Format**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Diagnostic | Output |

## 6-10.6 Mapping I/O Assembly Data Attribute Components

The following table indicates the I/O assembly Data attribute mapping for this inductive proximity switch device.

**Table 6-10.6 I/O Assembly Data Mapping**

| Data Component | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Diagnostic | Presence Sensing | 0Ehex | 1 | Diagnostic | 4 |
| Output | Presence Sensing | 0Ehex | 1 | Output | 1 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-10.7 Defining Device Configuration

Public access to the Presence Sensing Object by the Message Router must be supported for configuration of this device type. If supported, the optional Parameter Object may be used to access the device type's configuration parameter.

### 6-10.7.1 Parameter Object Instances

The following table identifies the Parameter Object instance supported by the inductive proximity switch.

**Table 6-10.7 Parameter Object Instances Supported**

| Number | Name |
|---|---|
| 1 | Operation Mode Configuration |

### 6-10.7.2 Mapping Parameter Object Data

The following table indicates the Parameter Object data mapping for the inductive proximity switch device.

**Table 6-10.8 Parameter Object Data Mapping**

| Configuration Parameter | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Operate Mode Configuration | Presence Sensing | 0Ehex | 1 | Operate Mode | 8 |

### 6-10.7.3 Configuration Parameter Definitions

The following sections of an example EDS show the information necessary to define the configuration parameters for an inductive proximity switch.

```
[Parms]
      Param1=                 $Operate Mode
      0,                      $Data Placeholder
      3,”20 0e 24 01 30 08”,  $Path size and Path to Operate Mode Attr
      0x0002,                 $Descriptor (support enumerated strings)
      0xC1,1                  $Data Type and Size (Boolean)
      ”Operate Mode”,         $Name
      ””,                     $Units (not used)
      ””,                     $User Manual Ref (not used)
      0,1,0,                  $min, max, default values
      0,0,0,0,                $mult, div, base, offset scaling (not used)
      0,0,0,0,                $mult, div, base, offset links (not used)
      1;                      $decimal places
[EnumPar]
      Param1=                 $Operate Mode Enumerated Strings
      ”Normally Open”,        $For value=0
      ”Normally Closed”;      $For value=1
```

## 6-10.8 Effect of Configuration Parameters on Behavior

The configuration parameter affects the device's behavior as shown below.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-10.9 Configuration Parameter Effect on Behavior**

| Parameter | Effect on Behavior |
|---|---|
| Operate Mode | Inverts the level defined for the Output attribute of the Presence Sensing Object |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-11 Photoelectric Sensor Device

## Device Type: 06 Hex

A photoelectric sensor electrically senses the presence or absence of a target object or part of a machine. Typical applications include assembly, packaging, and material handling.

## 6-11.1 Object Model

The Object Model in Figure 6-11.1 represents a photoelectric sensor. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

The CIP Object Library provides more details about these objects.

**Table 6-11.1 I/O Objects Present in a Photoelectric Sensor**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Required | 1 |
| Parameter | Optional | 1 |
| Presence Sensing | Required | 1 |

**Figure 6-11.1 Object Model for a Photoelectric Sensor**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-11.2 How Objects Affect Behavior

The objects for this device affect the device's behavior as shown in the table below.

**Table 6-11.2 Object Effect on Behavior**

| Object | Effect on behavior |
|---|---|
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines I/O data format |
| Parameter | Provides a public interface to the device's configuration data |
| Presence Sensing | Affects output value |

## 6-11.3 Defining Object Interfaces

The objects in this device have the interfaces listed in the following table:

**Table 6-11.3 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| Parameter | Message Router |
| Presence Sensing | Message Router, Assembly Object, or Parameter Object |

## 6-11.4 I/O Assembly Instances

The following table identifies the I/O assembly instance supported by the photoelectric sensor.

**Table 6-11.4 I/O Assembly Instances**

| Number | Type | Name |
|---|---|---|
| 1 | Input | Input Data |

## 6-11.5 I/O Assembly Data Attribute Format

The I/O Assembly data attribute has the format shown below.

**Table 6-11.5 I/O Assembly Data Attribute Format**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Diagnostic | Output |

## 6-11.6 Mapping I/O Assembly Data Attribute Components

The following table indicates the I/O assembly Data attribute mapping for this photoelectric sensor device.

**Table 6-11.6 I/O Assembly Data Mapping**

| Data Component | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Diagnostic | Presence Sensing | $0E_{hex}$ | 1 | Diagnostic | 4 |
| Output | Presence Sensing | $0E_{hex}$ | 1 | Output | 1 |

## 6-11.7 Defining Device Configuration

Public access to the Presence Sensing Object by the Message Router must be supported for configuration of this device type. If supported, the optional Parameter Object may be used to access the device type's configuration parameter.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-11.7.1 Parameter Object Instances

The photoelectric sensor contains one instance of the Parameter Object Class. This instance is a Parameter Object stub. See Chapter 5, Object Library, for the definition of the Parameter Object and an explanation of how it is used for configuration.

The following table identifies the Parameter Object instance supported by the photoelectric sensor.

**Table 6-11.7 Parameter Object Instances Supported**

| Number | Name |
|---|---|
| 1 | Operation Mode Configuration |

### 6-11.7.2 Mapping Parameter Object Data

The following table indicates the Parameter Object data mapping for the photoelectric sensor device.

**Table 6-11.8 Parameter Object Data Mapping**

| Configuration Parameter | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Operate Mode Configuration | Presence Sensing | $0E_{hex}$ | 1 | Operate Mode | 8 |

### 6-11.7.3 Configuration Parameter Definitions

The following sections of an example EDS show the information necessary to define the configuration parameters for photoelectric sensor.

```
[Parms]
      Param1=                $Operate Mode
      0,                     $Data Placeholder
      2,”20 0e 24 01 30 08”, $Path size and Path to Operate Mode Attr
      0x0002,                $Descriptor (support enumerated strings
      0xC1,1                 $Data Type and Size (Boolean)
      ”Operate Mode”,        $Name
      ””,                    $Units (not used)
      ””,                    $User Manual Ref (not used)
      0,1,0,                 $min, max, default values
      0,0,0,0,               $mult, div, base, offset scaling (not used)
      0,0,0,0,               $mult, div, base, offset links (not used)
      1;                     $decimal places
[EnumPar]
      Param1=                $Operate Mode Enumerated Strings
      ”Light Operate”,       $For value=0
      ”Dark Operate”;        $For value=1
```

## 6-11.8 Effect of Configuration Parameters on Behavior

The configuration parameter affects the device’s behavior as shown below.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-11.9 Configuration Parameter Effect on Behavior**

| Parameter | Effect on Behavior |
|---|---|
| Operate Mode | Inverts the level defined for the Output attribute of the Presence Sensing Object |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-12 General Purpose Discrete I/O Device

**Device Type: 07 Hex**

A General Purpose Discrete I/O device type interfaces to multiple discrete I/O device types that do not have network capabilities. Examples include sensors and actuators.

## 6-12.1 Object Model

The Object Model in Figure 6-12.1 represents a General Purpose Discrete I/O device. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

The CIP Object Library provides more details about these objects.

**Table 6-12.1 Objects Present in the General Purpose Discrete I/O Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Required | * |
| Discrete Input Group | Optional | 1 |
| Discrete Output Group | Optional | 1 |
| Discrete Input Point | ** | * |
| Discrete Output Point | *** | * |

* Depends on the level of I/O support provided by the product.
** Required for input functions
*** Required for output functions


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-12.1 Object Model for a General Purpose Discrete I/O Device**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-12.2 How Objects Affect Behavior

The objects for this device affect the device's behavior as shown in the table below.

**Table 6-12.2 Object Effect on Behavior**

| Object | Effect on behavior |
|---|---|
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines I/O data format and Output Configuration data format |
| Discrete Input Point | Defines behavior of the discrete input points for this device |
| Discrete Input Group | Stores the combined status of the Discrete Input Points |
| Discrete Output Point | Defines the behavior of discrete output points for this device |
| Discrete Output Group | Defines the Idle and Fault actions of the discrete output points |

## 6-12.3 Defining Object Interfaces

The objects in this device have the interfaces listed in the following table:

**Table 6-12.3 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| Discrete Input Group | Message Router |
| Discrete Output Group | Message Router |
| Discrete Input Point | Message Router |
| Discrete Output Point | Message Router or Assembly Object |

## 6-12.4 I/O Assembly Instances

The General Purpose Discrete I/O device I/O assemblies consist of:

- six predefined input assemblies with single input status bits
- one product-specific input assembly with a single input status bit
- six predefined input assemblies with multiple input status bits
- one product-specific input assembly with multiple input status bits
- six predefined output assemblies
- one product-specific output assembly
- six predefined input assemblies with output status bits
- one product-specific output status assembly
- four input assemblies with multiple input status bits and multiple output status bits
- nine input assemblies with a single input status bit and multiple output status bits


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The following table identifies the I/O assembly instances supported by this device.

**Table 6-12.4 I/O Assembly Instances**

| Number | Type | Name |
|---|---|---|
| 1 | Input | 1-Point Input with No Status Bit |
| 2 | Input | 2-Point Input with No Status Bit |
| 3 | Input | 4-Point Input with No Status Bit |
| 4 | Input | 8-Point Input with No Status Bit |
| 5 | Input | 16-Point Input with No Status Bit |
| 6 | Input | 32-Point Input with No Status Bit |
| 7 | Input | *N*-Point Input with No Status Bit |
| | | |
| 11 | Input | 1-Point Input with Single Status Bits |
| 12 | Input | 2-Point Input with Single Status Bit |
| 13 | Input | 4-Point Input with Single Status Bit |
| 14 | Input | 8-Point Input with Single Status Bit |
| 15 | Input | 16-Point Input with Single Status Bit |
| 16 | Input | 32-Point Input with Single Status Bit |
| 17 | Input | *N*-Point Input with Single Status Bit |
| | | |
| 21 | Input | 1-Point Input with Multiple Status Bits |
| 22 | Input | 2-Point Input with Multiple Status Bits |
| 23 | Input | 4-Point Input with Multiple Status Bits |
| 24 | Input | 8-Point Input with Multiple Status Bits |
| 25 | Input | 16-Point Input with Multiple Status Bits |
| 26 | Input | 32-Point Input with Multiple Status Bits |
| 27 | Input | *N*-Point Input with Multiple Status Bits |
| | | |
| 31 | Output | 1-Point Output |
| 32 | Output | 2-Point Output |
| 33 | Output | 4-Point Output |
| 34 | Output | 8-Point Output |
| 35 | Output | 16-Point Output |
| 36 | Output | 32-Point Output |
| 37 | Output | *N*-Point Output |
| | | |
| 41 | Input | 1-Point Output Status Bit |
| 42 | Input | 2-Point Output Status Bits |
| 43 | Input | 4-Point Output Status Bits |
| 44 | Input | 8-Point Output Status Bits |
| 45 | Input | 16-Point Output Status Bits |
| 46 | Input | 32-Point Output Status Bits |
| 47 | Input | *N*-Point Output Status Bits |
| | | |
| 52 | Input | 2-Point Input with Single Input Status and Single Output Status Bits |
| 53 | Input | 4-Point Input with Single Input Status and Single Output Status Bits |
| 54 | Input | 8-Point Input with Single Input Status and Single Output Status Bits |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Number | Type | Name |
|---|---|---|
| 55 | Input | 16-Point Input with Single Input Status and Single Output Status Bits |
| 56 | Input | 32-Point Input with Single Input Status and Single Output Status Bits |
| 57 | Input | *N*-Point Input with Single Input Status and Single Output Status Bits |
| 62 | Input | 2-Point Input with Multiple Input Status and Multiple Output Status Bits |
| 63 | Input | 4-Point Input with Multiple Input Status and Multiple Output Status Bits |
| 64 | Input | 8-Point Input with Multiple Input Status and Multiple Output Status Bits |
| 65 | Input | 16-Point Input with Multiple Input Status and Multiple Output Status Bits |
| 70 | Input | 1-Point Input with Single Input Status and 1 Output Status Bit |
| 71 | Input | 2-Point Input with Single Input Status and 1 Output Status Bit |
| 72 | Input | 2-Point Input with Single Input Status and 2 Output Status Bits |
| 73 | Input | 4-Point Input with Single Input Status and 2 Output Status Bits |
| 74 | Input | 4-Point Input with Single Input Status and 4 Output Status Bits |
| 75 | Input | 8-Point Input with Single Input Status and 4 Output Status Bits |
| 76 | Input | 8-Point Input with Single Input Status and 8 Output Status Bits |
| 77 | Input | 16-Point Input with Single Input Status and 8 Output Status Bits |
| 78 | Input | 16-Point Input with Single Input Status and 16 Output Status Bits |

## 6-12.5 I/O Assembly Data Attribute Format

The I/O Assembly data attribute for the input data with no status bit has the format shown below.

**Table 6-12.5 I/O Assembly Data Attribute Format – Instances w/no Status Bit**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | Reserved | | | | | | | Discrete Input1 |
| 2 | 0 | Reserved | | | | | | Discrete Input2 | Discrete Input 1 |
| 3 | 0 | Reserved | | | | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| 4 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| 5 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input16 | Discrete Input15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input9 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 6 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input16 | Discrete Input15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input9 |
| | 2 | Discrete Input24 | Discrete Input23 | Discrete Input22 | Discrete Input21 | Discrete Input20 | Discrete Input19 | Discrete Input18 | Discrete Input17 |
| | 3 | Discrete Input32 | Discrete Input31 | Discrete Input30 | Discrete Input29 | Discrete Input28 | Discrete Input27 | Discrete Input26 | Discrete Input25 |
| 7 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | •<br>•<br>• | | | | | | | | |
| | *M* | reserved | | | | | Discrete Input *N* | Discrete Input *N*-1 | Discrete Input *N*-2 |

The I/O Assembly data attribute for the input data with one status bit has the format shown below.

**Table 6-12.6 I/O Assembly Data Attribute Format – Instances w/one Status Bit**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 11 | 0 | Status | Reserved | | | | | | Discrete Input1 |
| 12 | 0 | Status | Reserved | | | | | Discrete Input2 | Discrete Input1 |
| 13 | 0 | Status | Reserved | | | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| 14 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Status | Reserved | | | | | | |
| 15 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input16 | Discrete Input15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input9 |
| | 2 | Status | Reserved | | | | | | |
| 16 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input16 | Discrete Input15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input9 |
| | 2 | Discrete Input24 | Discrete Input23 | Discrete Input22 | Discrete Input21 | Discrete Input20 | Discrete Input19 | Discrete Input18 | Discrete Input17 |
| | 3 | Discrete Input32 | Discrete Input31 | Discrete Input30 | Discrete Input29 | Discrete Input28 | Discrete Input27 | Discrete Input26 | Discrete Input25 |
| | 4 | Status | Reserved | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 17 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | •<br>•<br>• | | | | | | | | |
| | *M* | Status | Reserved | | | | Discrete Input *N* | Discrete Input *N*-1 | Discrete Input *N*-2 |

The I/O Assembly data attribute for the input data with multiple status bits has the format shown below.

**Table 6-12.7 I/O Assembly Data Attribute Format – Instances w/multiple Status Bits**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 22 | 0 | Reserved | | | | Status2 | Status1 | Discrete Input2 | Discrete Input1 |
| 23 | 0 | Status4 | Status3 | Status2 | Status1 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| 24 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Status8 | Status7 | Status6 | Status5 | Status4 | Status3 | Status2 | Status1 |
| 25 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input16 | Discrete Input15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input9 |
| | 2 | Status8 | Status7 | Status6 | Status5 | Status4 | Status3 | Status2 | Status1 |
| | 3 | Status16 | Status15 | Status14 | Status13 | Status12 | Status11 | Status10 | Status9 |
| 26 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | •<br>•<br>• | | | | | | | | |
| | 3 | Discrete Input32 | • | • | • | • | • | • | Discrete Input25 |
| | 4 | Status8 | • | • | • | • | • | • | Status1 |
| | •<br>•<br>• | | | | | | | | |
| | 7 | Status32 | • | • | • | • | • | • | Status25 |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 27 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | •<br>•<br>• | | | | | | | | |
| | *M*-2 | Status5 | Status4 | Status3 | Status2 | Status1 | Discrete Input *N* | Discrete Input *N*-1 | Discrete Input *N*-2 |
| | *M*-1 | • | • | • | • | • | • | • | Status6 |
| | *M* | Reserved | | | | | Status *N* | Status *N*-1 | Status *N*-2 |

The I/O Assembly data attribute for the output data has the format shown below.

**Table 6-12.8 I/O Assembly Data Attribute Format – Output Data Assembly**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 31 | 0 | Reserved | | | | | | | Discrete Output1 |
| 32 | 0 | Reserved | | | | | | Discrete Output2 | Discrete Output1 |
| 33 | 0 | Reserved | | | | Discrete Output4 | Discrete Output3 | Discrete Output2 | Discrete Output1 |
| 34 | 0 | Discrete Output8 | Discrete Output7 | Discrete Output6 | Discrete Output5 | Discrete Output4 | Discrete Output3 | Discrete Output2 | Discrete Output1 |
| 35 | 0 | Discrete Output8 | Discrete Output7 | Discrete Output6 | Discrete Output5 | Discrete Output4 | Discrete Output3 | Discrete Output2 | Discrete Output1 |
| | 1 | Discrete Output16 | Discrete Output15 | Discrete Output14 | Discrete Output13 | Discrete Output12 | Discrete Output11 | Discrete Output10 | Discrete Output9 |
| 36 | 0 | Discrete Output8 | Discrete Output7 | Discrete Output6 | Discrete Output5 | Discrete Output4 | Discrete Output3 | Discrete Output2 | Discrete Output1 |
| | 1 | Discrete Output16 | Discrete Output15 | Discrete Output14 | Discrete Output13 | Discrete Output12 | Discrete Output11 | Discrete Output10 | Discrete Output9 |
| | 2 | Discrete Output24 | Discrete Output23 | Discrete Output22 | Discrete Output21 | Discrete Output20 | Discrete Output19 | Discrete Output18 | Discrete Output17 |
| | 3 | Discrete Output32 | Discrete Output31 | Discrete Output30 | Discrete Output29 | Discrete Output28 | Discrete Output27 | Discrete Output26 | Discrete Output25 |
| 37 | 0 | Discrete Output8 | Discrete Output7 | Discrete Output6 | Discrete Output5 | Discrete Output4 | Discrete Output3 | Discrete Output2 | Discrete Output1 |
| | … | | | | | | | | |
| | *M* | reserved | | | | | Discrete Output *N* | Discrete Output *N*-1 | Discrete Output *N*-2 |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The I/O Assembly data attribute for the output data status bits has the format shown below.

**Table 6-12.9 I/O Assembly Data Attribute Format – Output Data Status Bits**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 41 | 0 | Reserved | | | | | | | Discrete Output Status1 |
| 42 | 0 | Reserved | | | | | | Discrete Output Status2 | Discrete Output Status1 |
| 43 | 0 | Reserved | | | | Discrete Output Status4 | Discrete Output Status3 | Discrete Output Status2 | Discrete Output Status1 |
| 44 | 0 | Discrete Output Status8 | Discrete Output Status7 | Discrete Output Status6 | Discrete Output Status5 | Discrete Output Status4 | Discrete Output Status3 | Discrete Output Status2 | Discrete Output Status1 |
| 45 | 0 | Discrete Output Status8 | Discrete Output Status7 | Discrete Output Status6 | Discrete Output Status5 | Discrete Output Status4 | Discrete Output Status3 | Discrete Output Status2 | Discrete Output Status1 |
| | 1 | Discrete Output Status16 | Discrete Output Status15 | Discrete Output Status14 | Discrete Output Status13 | Discrete Output Status12 | Discrete Output Status11 | Discrete Output Status10 | Discrete Output Status9 |
| 46 | 0 | Discrete Output Status8 | Discrete Output Status7 | Discrete Output Status6 | Discrete Output Status5 | Discrete Output Status4 | Discrete Output Status3 | Discrete Output Status2 | Discrete Output Status1 |
| | 1 | Discrete Output Status16 | Discrete Output Status15 | Discrete Output Status14 | Discrete Output Status13 | Discrete Output Status12 | Discrete Output Status11 | Discrete Output Status10 | Discrete Output Status9 |
| | 2 | Discrete Output Status24 | Discrete Output Status23 | Discrete Output Status22 | Discrete Output Status21 | Discrete Output Status20 | Discrete Output Status19 | Discrete Output Status18 | Discrete Output Status17 |
| | 3 | Discrete Output Status32 | Discrete Output Status31 | Discrete Output Status30 | Discrete Output Status29 | Discrete Output Status28 | Discrete Output Status27 | Discrete Output Status26 | Discrete Output Status25 |
| 47 | 0 | Discrete Output Status8 | Discrete Output Status7 | Discrete Output Status6 | Discrete Output Status5 | Discrete Output Status4 | Discrete Output Status3 | Discrete Output Status2 | Discrete Output Status1 |
| | … | | | | | | | | |
| | *M* | Reserved | | | | | Discrete Output Status *N* | Discrete Output Status *N*-1 | Discrete Output Status *N*-2 |

The I/O Assembly data attribute for the input data with one input status bit and one output status bit has the format shown below.

**Table 6-12.10 I/O Assembly Data Attribute Format – Assy with 1 Input & Output Status Bit**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 52 | 0 | Discrete Input Status | Discrete Output Status | Reserved | | | | Discrete Input2 | Discrete Input1 |
| 53 | 0 | Discrete Input Status | Discrete Output Status | Reserved | | Discrete Input4 | Discrete Input3 | Discrete Input 2 | Discrete Input1 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 54 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input Status | Discrete Output Status | Reserved | | | | | |
| 55 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input16 | Discrete Input15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input9 |
| | 2 | Discrete Input Status | Discrete Output Status | Reserved | | | | | |
| 56 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input16 | Discrete Input15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input9 |
| | 2 | Discrete Input24 | Discrete Input23 | Discrete Input22 | Discrete Input21 | Discrete Input20 | Discrete Input19 | Discrete Input18 | Discrete Input17 |
| | 3 | Discrete Input32 | Discrete Input31 | Discrete Input30 | Discrete Input29 | Discrete Input28 | Discrete Input27 | Discrete Input26 | Discrete Input25 |
| | 4 | Discrete Input Status | Discrete Output Status | Reserved | | | | | |
| 57 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | … | | | | | | | | |
| | *M* | Discrete Input Status | Discrete Output Status | Reserved | | | Discrete Input *N* | Discrete Input *N*-1 | Discrete Input *N*-2 |

The I/O Assembly data attribute for the input data with multiple input status bits and multiple output status bits has the format shown below.

**Table 6-12.11 I/O Assembly Data Attribute Format – Assy w/Multiple Input & Output Status Bits**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 62 | 0 | Reserved | | Discrete Output Status2 | Discrete Output Status1 | Discrete Input Status2 | Discrete Input Status1 | Discrete Input2 | Discrete Input1 |
| 63 | 0 | Discrete Input Status4 | Discrete Input Status3 | Discrete Input Status2 | Discrete Input Status1 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Reserved | | | | Discrete Output Status4 | Discrete Output Status3 | Discrete Output Status2 | Discrete Output Status1 |
| 64 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input Status8 | Discrete Input Status7 | Discrete Input Status6 | Discrete Input Sta6tus5 | Discrete Input Status4 | Discrete Input Status3 | Discrete Input Status2 | Discrete Input Status1 |
| | 2 | Discrete Output Status8 | Discrete Output Status7 | Discrete Output Status6 | Discrete Output Status5 | Discrete Output Status4 | Discrete Output Status3 | Discrete Output Status2 | Discrete Output Status1 |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 65 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input16 | Discrete Input15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input9 |
| | 2 | Discrete Input Status8 | Discrete Input Status7 | Discrete Input Status6 | Discrete Input Status5 | Discrete Input Status4 | Discrete Input Status3 | Discrete Input Status2 | Discrete Input Status1 |
| | 3 | Discrete Input Status16 | Discrete Input Status15 | Discrete Input Status14 | Discrete Input Status15 | Discrete Input Status12 | Discrete Input Status11 | Discrete Input Status10 | Discrete Input Status9 |
| | 4 | Discrete Output Status8 | Discrete Output Status7 | Discrete Output Status6 | Discrete Output Status5 | Discrete Output Status4 | Discrete Output Status3 | Discrete Output Status2 | Discrete Output Status1 |
| | 5 | Discrete Output Status16 | Discrete Output Status15 | Discrete Output Status14 | Discrete Output Status13 | Discrete Output Status12 | Discrete Output Status11 | Discrete Output Status10 | Discrete Output Status9 |

The I/O Assembly data attribute for the input data with single input status bit and multiple output status bits has the format shown below.

**Table 6-12.12 I/O Assembly Data Attribute Format –Assy w/Mixed Input & Output Status Bits**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 70 | 0 | Discrete Input Status | Reserved | | | | | Discrete Output Status1 | Discrete Input1 |
| 71 | 0 | Discrete Input Status | Reserved | | | | Discrete Output Status1 | Discrete Input2 | Discrete Input1 |
| 72 | 0 | Discrete Input Status | Reserved | | | Discrete Output Status2 | Discrete Output Status1 | Discrete Input2 | Discrete Input1 |
| 73 | 0 | Discrete Input Status | Reserved | Discrete Output Status2 | Discrete Output Status1 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| 74 | 0 | Discrete Output Status4 | Discrete Output Status3 | Discrete Output Status2 | Discrete Output Status1 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input Status | Reserved | | | | | | |
| 75 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input Status | Reserved | | | Discrete Output Status4 | Discrete Output Status3 | Discrete Output Status2 | Discrete Output Status1 |
| 76 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Output Status8 | Discrete Output Status7 | Discrete Output Status6 | Discrete Output Status5 | Discrete Output Status4 | Discrete Output Status3 | Discrete Output Status2 | Discrete Output Status1 |
| | 2 | Discrete Input Status | Reserved | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 77 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input16 | Discrete Input15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input9 |
| | 2 | Discrete Output Status8 | Discrete Output Status7 | Discrete Output Status6 | Discrete Output Status5 | Discrete Output Status4 | Discrete Output Status3 | Discrete Output Status2 | Discrete Output Status1 |
| | 3 | Discrete Input Status | Reserved | | | | | | |
| 78 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input16 | Discrete Input15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input9 |
| | 2 | Discrete Output Status8 | Discrete Output Status7 | Discrete Output Status6 | Discrete Output Status5 | Discrete Output Status4 | Discrete Output Status3 | Discrete Output Status2 | Discrete Output Status1 |
| | 3 | Discrete Output Status16 | Discrete Output Status15 | Discrete Output Status14 | Discrete Output Status13 | Discrete Output Status12 | Discrete Output Status11 | Discrete Output Status10 | Discrete Output Status9 |
| | 4 | Discrete Input Status | Reserved | | | | | | |

## 6-12.6 Mapping I/O Assembly Data Attribute Components

The following table indicates the I/O assembly Data attribute mapping for the General Purpose Discrete I/O device for the input assemblies with a single status bit.

**Table 6-12.13 I/O Assembly Data Mapping – Single Status Bit Assemblies**

| Data Component | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Discrete InputN | Discrete Input Point | 08hex | N | Value | 3 |
| Status or Discrete Input Status | Discrete Input Group | 1Dhex | 1 | Status | 5 |

**Important:** If I/O Assembly instances 7, 17, 27, 37, 47 or 57 are supported, the “Max Instance” attribute at the class level of the Discrete Input Point class or of the Discrete Output Point class must be supported.

The following table indicates the I/O assembly Data attribute mapping for the General Purpose Discrete I/O device for the input assemblies with multiple status bits.

**Table 6-12.14 I/O Assembly Data Mapping – Multiple Status Bit Assemblies**

| Data Component | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Discrete InputN | Discrete Input Point | $08_{hex}$ | N | Value | 3 |
| StatusN or Discrete Input StatusN | Discrete Input Point | $08_{hex}$ | N | Status | 4 |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The following table indicates the I/O assembly Data attribute mapping for the General Purpose Discrete I/O device for the output assemblies.

**Table 6-12.15 I/O Assembly Data Mapping – Output Assemblies**

| Data Component | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Discrete OutputN | Discrete Output Point | $09_{hex}$ | N | Value | 3 |
| Discrete Output StatusN | Discrete Output Point | $09_{hex}$ | N | Status | 4 |
| Discrete Output Status | Discrete Output Group | $1E_{hex}$ | 1 | Status | 5 |

## 6-12.7 Defining Device Configuration

Primary public interface to the Input Filter Selection parameter is accessed by the Discrete Output Group Object.

### 6-12.7.1 Input Configuration

There are no configuration parameters defined for the discrete inputs.

## 6-12.8 Output Configuration Assembly Instances

The following table identifies the output configuration assembly instance supported by the General Purpose Discrete I/O device.

**Table 6-12.16 Output Configuration Assembly Instances**

| Number | Type | Name |
|---|---|---|
| 40 | Configuration | Output Configuration |

## 6-12.9 Output Configuration Assembly Data Attribute Format

The Output Configuration Assembly Data attribute (typical throughout the document) has the format shown below.

**Table 6-12.17 Output Configuration Assembly Data Attribute Format**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 40 | 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Idle Action | Fault Action |

## 6-12.10 Mapping Output Configuration Assembly Data Attribute Components

The output configuration is accessed by instances of the Assembly Object Class. The following table indicates the output configuration assembly Data attribute mapping for the General Purpose Discrete I/O device.

**Table 6-12.18 Output Configuration Assembly Data Mapping**

| Configuration | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Parameter Name | Name | Number | Number | Name | Number |
| Fault Action | Discrete Output Group | $1E_{hex}$ | 1 | Fault Action | 7 |
| Idle Action | Discrete Output Group | $1E_{hex}$ | 1 | Idle Action | 9 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The following table shows the effect of the Fault State and Idle State parameters on behavior.

**Table 6-12.19 Effect of Fault State/Idle State on Behavior**

| Parameter | Effect on behavior |
|---|---|
| Fault Action | Indicates whether the Fault Value or the last state is to be placed at the output in the event of a fault. All Discrete Outputs in the device are set to the same Fault Action. **Note:** Fault Value cannot be configured via this assembly. The default is 0. |
| Idle Action | Indicates whether the Idle Value or the last state is to be placed at the output in the event of an idle. All Discrete Outputs in the device are set to the same Idle Action. **Note:** Idle Value cannot be configured via this assembly. The default is 0. |

The following portion of an example EDS shows the information necessary to define the output configuration parameters for the General Purpose Discrete I/O device.

```
[Params]
Param1=$ Fault Action
      0,                      $ reserved
      6,”20 1E 24 01 30 07”,  $ Link Path Size and Link Path
      0x0000,                 $ No support for settable path,
                              $ enumerated strings, scaling,
                              $ scaling link, or real time
                              $ update of value. Value is
                              $ gettable and settable.
      0xC1,                   $ Data Type
      1,                      $ Data Size
      ”Fault Action”,         $ Parameter Name
      ””,                     $ Units String not used
      ””,                     $ Help string not used
      0,1,0;                  $ Min, Max, and Default values
      1,1,1,0,0,0,0,0,0;      $ Not used
Param2=                       $ Idle Action
      0,                      $ reserved
      6,”20 1E 24 01 30 09”,  $ Link Path Size and Link Path
      0x00,                   $ No support for settable path,
                              $ enumerated strings, scaling,
                              $ scaling link, or real time
                              $ update of value. Value is
                              $ gettable and settable.
      4,                      $ Data Type
      1,                      $ Data Size
      ”Idle Action”,          $ Parameter Name
      ””,                     $ Units String not used
      ””,                     $ Help string not used
      0,1,0;                  $ Min, Max, and Default values
      1,1,1,0,0,0,0,0,0;      $ Not used
[Groups]                      $ No need to support
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-13 Communications Adapter Device

## Device Type: 0C Hex

The Communications Adapter device type acts as a gateway from the CIP network to other technologies. Traditionally, a gateway connects to foreign networks (for example, RS-232) or backplanes (for example, VME). The technologies involved greatly affect the gateway modeling and definition.

## 6-13.1 Object Model

The Object Model in Figure 6-13.1 represents the minimum support in a Communications Adapter. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-13.1 Objects Present in a Communications Adapter**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Optional | Possibly 1 or more |
| Application | Optional | Possibly 1 or more |
| Port | Conditional [1] | Possibly 1 or more |

1 Required if this device routes between two or more CIP subnets, optional otherwise

The Communications Adapter profile cannot specify the definition of the Assembly Object or the type of application objects necessary for device operation. This portion of the device profile must be supplied by the product developer as described in Sections 6-1 through 6-7 inclusive.

Since the Communications Adapter device profile does not define any static Assembly object instances, use of Assembly object instances in the “Open - Defined by the device profile” ranges (see Table 5A-5.2 Assembly Instance ID Ranges) for vendor specific purposes is permitted..

Application objects may be public, vendor-specific, or both.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-13.1 Object Model for the Communications Adapter**

1 - Required if this device routes between two or more CIP subnets, optional otherwise.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-14 General Purpose Analog I/O Device

The profile for this device type will be defined by the ODVA.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-15 AC/DC Drive Devices

**AC Drives Device Type: 02 Hex**
**DC Drives Device Type: 13 Hex**

These device profiles describe standard objects and behavior for AC and DC drives including Standard Scalar (V/Hz) AC, Vector AC, and DC Drives.

The functionality of drives covered includes:

- Open loop speed (frequency) control
- Closed loop speed (frequency) control
- Torque control
- No position control

These profiles make the drives inter-operable, but not directly interchangeable without doing drive configuration through the drive local interface, a network configuration tool or other means of configuration outside the CIP interface.

The AC and DC Drive profiles are part of a “Hierarchy of Motor Control Devices” that is supported by CIP. This hierarchy includes:

- Contactors and Across the Line Motor Starters
- Soft Starters
- AC and DC Drives
- Servo Drives

Devices within this hierarchy all use a common “Control Supervisor” object to control the state behavior of the device. All but the low level Contactors and Across the Line Motor Starters also use a common “Motor Data” object to store information about the motor to be controlled. The Hierarchy of Motor Control Devices also supports a hierarchy of IO Assembly Instance definitions. Assembly instances are numbered within the hierarchy so that each device type is assigned a range of Assembly Instance numbers, with higher functionality devices supporting higher instance numbers. Devices in the hierarchy can choose to support some IO Assembly Instance numbers that are lower than theirs in the hierarchy. For example an AC Drive may choose to support some IO Assemblies that are defined in the Starter Profile to make it easier to interchange drives and starters in a system.

## 6-15.1 Multiple axes on one drive

It is possible to implement several axes of control on one physical drive unit. A separate MAC ID must be assigned to each axis so each axis is treated as separate CIP node.

## 6-15.2 Object Model

The Object Model in Figure 6-15.1 represents an AC or DC Drive. The table below indicates

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Chapter 5, The CIP Object Library, provides more details about these objects.

**Table 6-15.1 Objects Present in an AC/DC Drive**

| Object Class | Optional / Required | # of Instances |
|---|---|---|
| Message Router | Optional | 1 |
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Required | 2 |
| Control Supervisor | Required | 1 |
| AC/DC Drive | Required | 1 |
| Motor Data | Required | 1 |
| Parameter | Optional | - |
| Parameter Group | Optional | - |
| Discrete Input | Optional | - |
| Discrete Output | Optional | - |
| Analog Input | Optional | - |
| Analog Output | Optional | - |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-15.1 Object Model for AC and DC Drives**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-15.3 How Objects Affect Behavior

The objects in this device affect the device's behavior as shown in the following table.

**Table 6-15.2 Object Effect on Behavior**

| Object | Effect on Behavior |
|---|---|
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines I/O data format |
| Control Supervisor | Manages drive functions, operational states and control |
| AC/DC Drive | Provides drive configuration |
| Motor Data | Defines motor data for the motor connected to this device |
| Parameter | Provides a public interface to device configuration data |
| Parameter Group | Provides an aid to device configuration |
| Discrete Input | Defines the behavior of discrete inputs on this device |
| Discrete Output | Defines the behavior of discrete outputs on this device |
| Analog Input | Defines the behavior of analog inputs on this device |
| Analog Output | Defines the behavior of analog outputs on this device |

## 6-15.4 Defining Object Interfaces

The objects in this device have the interfaces listed in the following table.

**Table 6-15.3 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| Control Supervisor | Message Router, Assembly or Parameter Object |
| AC/DC Drive | Message Router, Assembly or Parameter Object |
| Motor Data | Message Router or Parameter Object |
| Parameter | Message Router |
| Discrete Input | Message Router or Assembly |
| Discrete Output | Message Router or Assembly |
| Analog Input | Message Router or Assembly |
| Analog Output | Message Router or Assembly |

## 6-15.5 I/O Assembly Instances

The IO Assembly Instance definitions in this section define the format of the “data” attribute (attribute 3) for IO Assembly Instances. Through the use of predefined instance definitions, IO Assemblies support a hierarchy of motor control devices. The device hierarchy includes motor starters, soft starters, AC and DC drives, and servo drives. Assembly Instances are numbered within the hierarchy so that each device type is assigned a range of Assembly Instance numbers, with higher functionality devices supporting higher instance numbers. **Devices in the hierarchy can choose to support instance numbers that are lower than theirs in the hierarchy.** For example an AC drive may choose to support some IO Assemblies that are defined in the starter profile to make it easier to interchange starters and drives within a system. The following table shows the Assembly Instance numbering for the motor control device hierarchy.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-15.4 I/O Assembly Instance Numbering Ranges**

| Profile | I/O Type | Instance Range | Instances within hierarchy that may be implemented for this product type. |
|---|---|---|---|
| AC Motor Starter | Output | 1-19 | 1-19 |
| Soft Start Starter | Input | 50-69 | 50-69 |
| AC or DC Drive | Output | 20-29 | 1-29 |
| | Input | 70-79 | 50-79 |
| Servo Drive | Output | 30-49 | 1-49 |
| | Input | 80-99 | 50-99 |

The following IO Assembly Instances are defined for AC and DC Drives.

**Table 6-15.5 I/O Assembly Instances**

| Number | | | | |
|---|---|---|---|---|
| **decimal** | **hex** | **Required/Optional** | **Type** | **Name** |
| 20 | 14 | Required | Output | Basic Speed Control Output |
| 21 | 15 | Optional | Output | Extended Speed Control Output |
| 22 | 16 | Optional | Output | Speed and Torque Control Output |
| 23 | 17 | Optional | Output | Extended Speed and Torque Control Output |
| 24 | 18 | Optional | Output | Process Control Output |
| 25 | 19 | Optional | Output | Extended Process Control Output |
| | | | | |
| 70 | 46 | Required | Input | Basic Speed Control Input |
| 71 | 47 | Optional | Input | Extended Speed Control Input |
| 72 | 48 | Optional | Input | Speed and Torque Control Input |
| 73 | 49 | Optional | Input | Extended Speed and Torque Control Input |
| 74 | 4A | Optional | Input | Process Control Input |
| 75 | 4B | Optional | Input | Extended Process Control Input |

If a bit is not used in an IO Assembly, it is reserved for use in other Assemblies. Reserved bits in Output Assemblies are ignored by the consuming device. Reserved bits in Input Assemblies are set to zero by the producing device.

Reserved bits in the IO Assembly Data Attribute Format Tables are shaded.

### 6-15.5.1 Connection Paths to I/O Assembly Instances

The IO Assembly Instances are chosen for IO Connections by setting the “produced_connection_path” (attribute 14) and “consumed_connection_path” (attribute 16) attributes in the appropriate connection object.

AC and DC Drives use the Symbolic Segment Type (see Appendix C) to specify paths to the IO Assembly Instances in the Motor Control Hierarchy. IO Assembly Instances are represented by ASCII strings that contain the hex number of the Assembly Instance whose path is to be chosen.

The following example shows the Symbolic Segment used to specify Output Assembly Instance 20 (14 hex).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-15.2 Symbolic Segment Example for Output Assy**

| Segment Type | | First Character | Second Character |
|---|---|---|---|
| 0 1 1 | 0 0 0 1 0 | 0 0 1 1 0 0 0 1 | 0 0 1 1 0 1 0 0 |
| Segment type (symbolic) | Symbol size in bytes (2 bytes) | ASCII 1 (31 hex) | ASCII 4 (34 hex) |

## 6-15.6 I/O Assembly Data Attribute Format

The I/O Assembly Data Attributes have the format shown below.

**Table 6-15.6 I/O Assembly Data Attribute Format**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 0 | | | | | | Fault Reset | | Run Fwd |
| | 1 | | | | | | | | |
| | 2 | Speed Reference (Low Byte) | | | | | | | |
| | 3 | Speed Reference (High Byte) | | | | | | | |
| 21 | 0 | | NetRef | NetCtrl | | | Fault Reset | Run Rev | Run Fwd |
| | 1 | | | | | | | | |
| | 2 | Speed Reference (Low Byte) | | | | | | | |
| | 3 | Speed Reference (High Byte) | | | | | | | |
| 22 | 0 | | | | | | Fault Reset | | Run Fwd |
| | 1 | | | | | | | | |
| | 2 | Speed Reference (Low Byte) | | | | | | | |
| | 3 | Speed Reference (High Byte) | | | | | | | |
| | 4 | Torque Reference (Low Byte) | | | | | | | |
| | 5 | Torque Reference (High Byte) | | | | | | | |
| 23 | 0 | | NetRef | NetCtrl | | | Fault Reset | Run Rev | Run Fwd |
| | 1 | | | | | | | | |
| | 2 | Speed Reference (Low Byte) | | | | | | | |
| | 3 | Speed Reference (High Byte) | | | | | | | |
| | 4 | Torque Reference (Low Byte) | | | | | | | |
| | 5 | Torque Reference (High Byte) | | | | | | | |
| 24 | 0 | | | | | | Fault Reset | | Run Fwd |
| | 1 | | | | | | | | |
| | 2 | Speed Reference (Low Byte) | | | | | | | |
| | 3 | Speed Reference (High Byte) | | | | | | | |
| | 4 | Process Reference (Low Byte) | | | | | | | |
| | 5 | Process Reference (High Byte) | | | | | | | |
| 25 | 0 | NetProc | NetRef | NetCtrl | | | Fault Reset | Run Rev | Run Fwd |
| | 1 | Mode | | | | | | | |
| | 2 | Speed Reference (Low Byte) | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## AC/DC Drive Devices, Types: $02_{Hex}$ and $13_{Hex}$

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| | 3 | Speed Reference (High Byte) | | | | | | | |
| | 4 | Process Reference (Low Byte) | | | | | | | |
| | 5 | Process Reference (High Byte) | | | | | | | |
| 70 | 0 | | | | | | Running1 | | Faulted |
| | 1 | | | | | | | | |
| | 2 | Speed Actual (Low Byte) | | | | | | | |
| | 3 | Speed Actual (High Byte) | | | | | | | |
| 71 | 0 | At Reference | Ref From Net | Ctrl From Net | Ready | Running2 (Rev) | Running1 (Fwd) | Warning | Faulted |
| | 1 | Drive State | | | | | | | |
| | 2 | Speed Actual (Low Byte) | | | | | | | |
| | 3 | Speed Actual (High Byte) | | | | | | | |
| 72 | 0 | | | | | | Running1 | | Faulted |
| | 1 | | | | | | | | |
| | 2 | Speed Actual (Low Byte) | | | | | | | |
| | 3 | Speed Actual (High Byte) | | | | | | | |
| | 4 | Torque Actual (Low Byte) | | | | | | | |
| | 5 | Torque Actual (High Byte) | | | | | | | |
| 73 | 0 | At Reference | Ref From Net | Ctrl From Net | Ready | Running2 (Rev) | Running1 (Fwd) | Warning | Faulted |
| | 1 | Drive State | | | | | | | |
| | 2 | Speed Actual (Low Byte) | | | | | | | |
| | 3 | Speed Actual (High Byte) | | | | | | | |
| | 4 | Torque Actual (Low Byte) | | | | | | | |
| | 5 | Torque Actual (High Byte) | | | | | | | |
| 74 | 0 | | | | | | Running1 | | Faulted |
| | 1 | | | | | | | | |
| | 2 | Speed Actual (Low Byte) | | | | | | | |
| | 3 | Speed Actual (High Byte) | | | | | | | |
| | 4 | Process Actual (Low Byte) | | | | | | | |
| | 5 | Process Actual (High Byte) | | | | | | | |
| 75 | 0 | At Reference | Ref From Net | Ctrl From Net | Ready | Running2 (Rev) | Running1 (Fwd) | Warning | Faulted |
| | 1 | Drive State | | | | | | | |
| | 2 | Speed Actual (Low Byte) | | | | | | | |
| | 3 | Speed Actual (High Byte) | | | | | | | |
| | 4 | Process Actual (Low Byte) | | | | | | | |
| | 5 | Process Actual (High Byte) | | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-15.7 Mapping I/O Assembly Data Attribute Components

The following table indicates the I/O Assembly Data Attribute mapping for AC and DC Drive Output Assemblies.

**Table 6-15.7 I/O Assembly Data Mapping – Output Assemblies**

| Data Component | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| RunFwd | Control Supervisor | $29_{hex}$ | 1 | Run1 | 3 |
| RunRev | Control Supervisor | $29_{hex}$ | 1 | Run2 | 4 |
| Fault Reset | Control Supervisor | $29_{hex}$ | 1 | FaultRst | 12 |
| NetCtrl | Control Supervisor | $29_{hex}$ | 1 | NetCtrl | 5 |
| NetRef | AC/DC Drive | $2A_{hex}$ | 1 | NetRef | 4 |
| Net Proc | AC/DC Drive | $2A_{hex}$ | 1 | NetProc | 5 |
| Drive Mode | AC/DC Drive | $2A_{hex}$ | 1 | DriveMode | 6 |
| Speed Reference | AC/DC Drive | $2A_{hex}$ | 1 | SpeedRef | 8 |
| Torque Reference | AC/DC Drive | $2A_{hex}$ | 1 | TorqueRef | 12 |
| Process Reference | AC/DC Drive | $2A_{hex}$ | 1 | ProcessRef | 14 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The following table indicates the I/O Assembly Data Attribute mapping for AC and DC Drive Input Assemblies.

**Table 6-15.8 I/O Assembly Data Mapping – Input Assemblies**

| Data Component | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Faulted | Control Supervisor | $29_{hex}$ | 1 | Faulted | 10 |
| Warning | Control Supervisor | $29_{hex}$ | 1 | Warning | 11 |
| Running1 (Fwd) | Control Supervisor | $29_{hex}$ | 1 | Running1 | 7 |
| Running2 (Rev) | Control Supervisor | $29_{hex}$ | 1 | Running2 | 8 |
| Ready | Control Supervisor | $29_{hex}$ | 1 | Ready | 9 |
| CtrlFromNet | Control Supervisor | $29_{hex}$ | 1 | CtrlFromNet | 15 |
| Drive State | Control Supervisor | $29_{hex}$ | 1 | State | 6 |
| Ref From Net | AC/DC Drive | $2A_{hex}$ | 1 | RefFromNet | 29 |
| At Reference | AC/DC Drive | $2A_{hex}$ | 1 | AtReference | 3 |
| Speed Actual | AC/DC Drive | $2A_{hex}$ | 1 | SpeedActual | 7 |
| Torque Actual | AC/DC Drive | $2A_{hex}$ | 1 | TorqueActual | 11 |
| Process Actual | AC/DC Drive | $2A_{hex}$ | 1 | ProcessActual | 13 |

## 6-15.8 Defining Device Configuration

Public access to the Control Supervisor Object, the Motor Data Object, and the AC/DC Drive Object must be supported for configuration of an AC or DC drive. If supported, the optional Parameter Objects may be used to access the various configuration attributes in the Control Supervisor Object, the Motor Data Object, and the AC/DC Drive Object.

AC and DC drives may contain (but are not limited to) any of the Parameter Object instances listed in the table below. Suggested parameter names are also given in the table. The set of parameters instances that are supported by a drive should be numbered sequentially with lower instance numbers assigned to parameters that appear earlier in the table. Vendor specific parameter instances should be numbered sequentially following the instances that appear in the following table.

Parameter Object instances may be implemented as EDS file definitions, parameter stubs, or full parameter objects. See Chapter 5 for a definition of the Parameter Object and an explanation of how it is used for configuration.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-15.8.1 Mapping Parameter Object Data

The following table indicates the Parameter Object data mapping for an AC or DC Drive device.

**Table 6-15.9 Parameter Object Data Mapping**

| Configuration Parameter | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Motor Type | Motor Data | $28_{hex}$ | 1 | MotorType | 3 |
| Motor Cat Number | Motor Data | $28_{hex}$ | 1 | CatNumber | 4 |
| Motor Vendor | Motor Data | $28_{hex}$ | 1 | Manufacturer | 5 |
| Motor Rated Cur | Motor Data | $28_{hex}$ | 1 | RatedCurrent | 6 |
| Motor Rated Volt | Motor Data | $28_{hex}$ | 1 | RatedVoltage | 7 |
| Motor Rated Pwr | Motor Data | $28_{hex}$ | 1 | RatedPower | 8 |
| Motor Rated Freq | Motor Data | $28_{hex}$ | 1 | RatedFreq | 9 |
| Motor Rated Temp | Motor Data | $28_{hex}$ | 1 | RatedTemp | 10 |
| Motor Max Speed | Motor Data | $28_{hex}$ | 1 | MaxSpeed | 11 |
| Motor Pole Count | Motor Data | $28_{hex}$ | 1 | PoleCount | 12 |
| Motor Torq Const | Motor Data | $28_{hex}$ | 1 | TorqConstant | 13 |
| Motor Inertia | Motor Data | $28_{hex}$ | 1 | Inertia | 14 |
| Motor Base Speed | Motor Data | $28_{hex}$ | 1 | BaseSpeed | 15 |
| Motor Field Cur | Motor Data | $28_{hex}$ | 1 | RatedFieldCur | 16 |
| Min Field Cur | Motor Data | $28_{hex}$ | 1 | MinFieldCur | 17 |
| Rated Field Volt | Motor Data | $28_{hex}$ | 1 | RatedFieldVolt | 18 |
| Service Factor | Motor Data | $28_{hex}$ | 1 | ServiceFactor | 19 |
| Network Control | Control Supervisor | $29_{hex}$ | 1 | NetCtrl | 5 |
| Drive State | Control Supervisor | $29_{hex}$ | 1 | State | 6 |
| Running Fwd | Control Supervisor | $29_{hex}$ | 1 | Running1 | 7 |
| Running Rev | Control Supervisor | $29_{hex}$ | 1 | Running2 | 8 |
| Ready | Control Supervisor | $29_{hex}$ | 1 | Ready | 9 |
| Faulted | Control Supervisor | $29_{hex}$ | 1 | Faulted | 10 |
| Warning | Control Supervisor | $29_{hex}$ | 1 | Warning | 11 |
| Fault Reset | Control Supervisor | $29_{hex}$ | 1 | FaultRst | 12 |
| Fault Code | Control Supervisor | $29_{hex}$ | 1 | FaultCode | 13 |
| Warning Code | Control Supervisor | $29_{hex}$ | 1 | WarningCode | 14 |
| Control From Net | Control Supervisor | $29_{hex}$ | 1 | CtrlFromNet | 15 |
| DN Fault Mode | Control Supervisor | $29_{hex}$ | 1 | DNFaultMode | 16 |
| Force Fault | Control Supervisor | $29_{hex}$ | 1 | ForceFault/Trip | 17 |
| Force Status | Control Supervisor | $29_{hex}$ | 1 | ForceStatus | 18 |
| At Reference | AC/DC Drive | $2A_{hex}$ | 1 | AtReference | 3 |
| Network Ref | AC/DC Drive | $2A_{hex}$ | 1 | NetRef | 4 |
| Network Process | AC/DC Drive | $2A_{hex}$ | 1 | NetProc | 5 |
| Drive Mode | AC/DC Drive | $2A_{hex}$ | 1 | DriveMode | 6 |
| Speed Actual | AC/DC Drive | $2A_{hex}$ | 1 | SpeedActual | 7 |
| Speed Reference | AC/DC Drive | $2A_{hex}$ | 1 | SpeedRef | 8 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Configuration Parameter | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Current Actual | AC/DC Drive | $2A_{hex}$ | 1 | CurrentActual | 9 |
| Current Limit | AC/DC Drive | $2A_{hex}$ | 1 | CurrentLimit | 10 |
| Torque Actual | AC/DC Drive | $2A_{hex}$ | 1 | TorqueActual | 11 |
| Torque Reference | AC/DC Drive | $2A_{hex}$ | 1 | TorqueRef | 12 |
| Process Actual | AC/DC Drive | $2A_{hex}$ | 1 | ProcessActual | 13 |
| Process Reference | AC/DC Drive | $2A_{hex}$ | 1 | ProcessRef | 14 |
| Power Actual | AC/DC Drive | $2A_{hex}$ | 1 | PowerActual | 15 |
| Input Voltage | AC/DC Drive | $2A_{hex}$ | 1 | InputVoltage | 16 |
| Output Voltage | AC/DC Drive | $2A_{hex}$ | 1 | OutputVoltage | 17 |
| Accel Time | AC/DC Drive | $2A_{hex}$ | 1 | AccelTime | 18 |
| Decel Time | AC/DC Drive | $2A_{hex}$ | 1 | DecelTime | 19 |
| Low Speed Limit | AC/DC Drive | $2A_{hex}$ | 1 | LowSpdLimit | 20 |
| High Speed Limit | AC/DC Drive | $2A_{hex}$ | 1 | HighSpdLimit | 21 |
| Speed Scale | AC/DC Drive | $2A_{hex}$ | 1 | SpeedScale | 22 |
| Current Scale | AC/DC Drive | $2A_{hex}$ | 1 | CurrentScale | 23 |
| Torque Scale | AC/DC Drive | $2A_{hex}$ | 1 | TorqueScale | 24 |
| Process Scale | AC/DC Drive | $2A_{hex}$ | 1 | ProcessScale | 25 |
| Power Scale | AC/DC Drive | $2A_{hex}$ | 1 | PowerScale | 26 |
| Voltage Scale | AC/DC Drive | $2A_{hex}$ | 1 | VoltageScale | 27 |
| Time Scale | AC/DC Drive | $2A_{hex}$ | 1 | TimeScale | 28 |
| Ref From Net | AC/DC Drive | $2A_{hex}$ | 1 | RefFromNet | 29 |
| Proc From Net | AC/DC Drive | $2A_{hex}$ | 1 | ProcFromNet | 30 |
| Field I or V | AC/DC Drive | $2A_{hex}$ | 1 | FieldIorV | 31 |
| Field Voltage Ratio | AC/DC Drive | $2A_{hex}$ | 1 | FieldVoltRatio | 32 |
| Field Cur Set Pt | AC/DC Drive | $2A_{hex}$ | 1 | FieldCurSetPt | 33 |
| Field Weak Ena | AC/DC Drive | $2A_{hex}$ | 1 | FieldWkEnable | 34 |
| Field Cur Actual | AC/DC Drive | $2A_{hex}$ | 1 | FieldCurActual | 35 |
| Field Min Cur | AC/DC Drive | $2A_{hex}$ | 1 | FieldMinCur | 36 |

### 6-15.8.2 Parameter Group Objects

AC and DC drives may contain (but are not limited to) any of the Parameter Group Object Instances listed in the table below. If Parameter Groups are supported, Parameter Instances should be grouped according to the object that their data is mapped to. For example, all Parameters Instances whose data maps to the Motor Data Object should be contained in the Motor Group (Parameter Group Object Instance 1).

Parameter Group Object instances may be implemented from an EDS file, or as actual Parameter Group objects from the device. See Chapter 5 for a definition of the Parameter Group Object.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-15.10 Parameter Group Object Instances**

| Parameter Group Name | Instance Number | Parameters in Group |
|---|---|---|
| Motor | 1 | Motor Type<br>Motor Cat Number<br>Motor Vendor<br>Motor Rated Cur<br>Motor Rated Volt<br>Motor Rated Pwr<br>Motor Rated Freq<br>Motor Rated Temp<br>Motor Max Speed<br>Motor Pole Count<br>Motor Torq Const<br>Motor Inertia<br>Motor Base Speed<br>Motor Field Cur<br>Min Field Cur<br>Rated Field Volt<br>Rated Field Volt<br>Service Factor |
| Supervisor | 2 | Network Control<br>Drive State<br>Running Fwd<br>Running Rev<br>Ready<br>Faulted<br>Warning<br>Fault Reset<br>Fault Code<br>Warning Code<br>Control From Net<br>DN Fault Mode<br>Force Fault<br>Force Status |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Parameter Group Name | Instance Number | Parameters in Group |
|---|---|---|
| Drive | 3 | At Reference<br>Network Ref<br>Network Process<br>Drive Mode<br>Speed Actual<br>Speed Reference<br>Current Actual<br>Current Limit<br>Torque Actual<br>Torque Reference<br>Process Actual<br>Process Reference<br>Power Actual<br>Input Voltage<br>Output Voltage<br>Accel Time<br>Decel Time<br>Low Speed Limit<br>High Speed Limit<br>Speed Scale<br>Current Scale<br>Torque Scale<br>Process Scale<br>Power Scale<br>Voltage Scale<br>Time Scale<br>Ref From Net<br>Proc From Net<br>Field I or V<br>Field Voltage Ratio<br>Field Cur Set Pt<br>Field Weak Ena<br>Field Cur Actual<br>Field Min Cur |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-16 Servo Drive Device

This device profile for a Servo Drive is presently under development. It is intended that this profile will be defined by the Servo Drive SIG of ODVA.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-17 Barcode Scanner Device

The profile for this device type will be defined by the ODVA.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-18 Position Controller Device

**Device Type: 10hex**

A Position Controller controls the motion and position of a motor or linear actuator (servo, stepper, etc.). The position controller may or may not include an integrated drive. Positioning is achieved with a controlled motion profile. The motion profile is defined by Acceleration, Velocity, Deceleration, Position or Torque.

## 6-18.1 Object Model

The Object Model in Figure 6-18.1 represents a Position Controller Device. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

Chapter 5, Object Library, provides more details about these objects.

**Table 6-18.1 Objects Present in a Position Controller Device**

| Object Class | Optional / Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Position Controller Supervisor | Required | 1 per Axis |
| Position Controller | Required | 1 per Axis |
| Command Block | Optional | - |
| Block Sequencer | Optional | 1 per Axis |
| Drive | Optional | 1 per Axis |
| Motor Data | Optional | 1 per Axis |
| Parameter | Optional | - |

### 6-18.1.1 Model Description

The object model shown below describes how the Position Controller device is controlled through CIP. Attributes can be set and queried in the normal manner for configuration. The Position Controller object handles the interface to the internal or external drive unit. The motor and drive units can be servo, stepper or some other method with optional feedback (open or closed loop).

In addition, the Command Block object and the Block Sequencer object can be used to perform complex moves, modify attributes or wait for attributes to become valid. Command Blocks can be linked together to form a command chain with branching and looping supported. The user can download command block sequences during configuration and execute them at any time to perform complex moves or modify attributes. The Block Sequencer object is accessible through the I/O command message giving the user the ability to perform complex motion sequences from a PLC or scanner card.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-18.1 Object Model for a Position Controller**

## 6-18.2 How Objects Affect Behavior

The objects for this device affect the device's behavior as shown in the table below.

**Table 6-18.2 Object Effect on Behavior**

| Object | Effect on behavior |
|---|---|
| CIP Required Objects | See Section 6-2.2 for details. |
| Position Controller Supervisor | Handles faults, home and registration inputs and modifies meaning of I/O data |
| Position Controller | Provides positioning control and manages interface to power amplifier |
| Block Sequencer | Executes command block sequences |
| Command Block | Defines the behavior of command blocks |
| Drive | Manages power amplifier |
| Motor Data | Configures the power amplifier for motor parameters |
| Parameter | Defines the behavior of Parameters |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-18.3 Defining Object Interfaces

The objects in this device have the interfaces listed in the following table:

**Table 6-18.3 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Position Controller Supervisor | Message Router or Position Controller Supervisor Class |
| Position Controller | Message Router or Position Controller Supervisor Class |
| Block Sequencer | Message Router or Position Controller Supervisor Class |
| Command Block | Message Router or Position Controller Supervisor Class |
| Drive | Message Router or Position Controller Supervisor Class |
| Motor Data | Message Router or Position Controller Supervisor Class |
| Parameter | Message Router |

## 6-18.4 I/O Connection Messages

The Position Controller Profile supports both command and response messages via the I/O connection. The produced and consumed paths specify the Position Controller Supervisor Class attributes as shown in Figure 6-18.1.

### 6-18.4.1 Message Formats

The Position Controller Profile supports multiple axes per CIP node by allowing up to seven instances of each of the axis objects, in one Position Controller device. The axis objects are the 1) position controller supervisor, 2) position controller: 3) drive: 4) motor data: and 5) block sequencer. The I/O message can contain data from more than one axis object. The Command Axis Number and the Response Axis Number shown in the Message Format specifies the instance number of the axis object whose data is contained in the I/O message.

**Table 6-18.4 Command Message Format**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Arm | Hard Stop | Smooth Stop | Direction (V. Mode) | Incremental | Start Block | Load Data/ Start Profile |
| 1 | Command Data 1 | | | | | | | |
| | | | | | | | | |
| 2 | Command Axis Number | | | Command Message Type | | | | |
| 3 | Command Data 2 | | | | | | | |
| 4 | Command Data 3 | | | | | | | |
| 5 | Command Data 4 | | | | | | | |
| 6 | Command Data 5 | | | | | | | |
| 7 | Command Data 6 | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-18.5 Response Message Format**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Response Data 1 | | | | | | | |
| 2 | Load Complete | Block Fault | FE Fault | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Response Data 2 | | | | | | | |
| 5 | Response Data 3 | | | | | | | |
| 6 | Response Data 4 | | | | | | | |
| 7 | Response Data 5 | | | | | | | |

Note that a response message may contain data for a different axis number from what was contained in the command message. If an error is detected in the command or its requested response message, the response message shall be Command/Response Error Message Type 14 hex, not the requested response message.

### 6-18.4.2 Definition of a Profile Move

A profile move is a move that uses Acceleration, Target Velocity, and Deceleration to run at a Target Velocity or to a Target Position. In addition, the position controller device can output a Torque command. Whether or not the position controller device runs at a Target Velocity, to a Target Position or outputs a Torque command depends on the Operating Mode (Position Controller Object Attribute 3), to which the position controller device is set. The position controller device is set to Position, Velocity or Torque Mode using Position Controller Object Attribute 3.

### 6-18.4.3 Starting a Profile Move

The Position Controller Profile is mode-sensitive. The Position Controller Object Attribute 3 sets the mode of the controller to the following:

0 = Position (default)

1 = Velocity

2 = Torque

A profile move starts when the command message type for the specified mode is loaded and the Load Data/Start Profile bit transitions from zero to one. The table below shows the command message type, which starts a profile move for each mode.

**Table 6-18.6 Command Message Type**

| Mode (Attribute 3) | Command Message Type which Starts Motion |
|---|---|
| 0 = Position | 01 = Position |
| 1 = Velocity | 02 = Velocity |
| 2 = Torque | 05 = Torque |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-18.4.4 I/O Handshaking Procedure

Proper handshaking between the client and the server is essential in I/O messaging to ensure that data sent to the position controller device is properly received. Two bits are used to provide handshaking between command and response messages. The recommended handshaking procedure for the client is described in the flowcharts in Figure 6-18.2 and Figure 6-18.3 below. The behavior required from the server to implement the handshake procedure is described in Figure 6-18.4 and Figure 6-18.5. Refer to the timing diagram in Figure 6-18.6 below for representative timing of these bits during the handshake sequence

**Figure 6-18.2 Client Data Loading Procedure**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-18.3 Client Profile Move Procedure**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-18.4 Server Behavior**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-18.5 Profile Monitoring Procedure**

**Figure 6-18.6 Timing Diagram**

***** Data Successfully Loaded into Position Control Object


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-18.5 I/O Connection Message Types

### 6-18.5.1 Command Message Types

The command message type is defined by byte 02 of the message. Byte 00 is the same for all command message types. Bytes 01 and 03 through 07 are defined by the Command Message Type code in byte 02. In message types 01 through 05, byte 03 defines the requested Response axis number and Response Message Type format. For message types 19 hex through 1F hex, the requested Response axis number and Response Message Type is the same as the Command axis number and Command Message Type.

**Table 6-18.7 Command Message Type 01 hex Target Position - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Arm | Hard Stop | Smooth Stop | Direction (V. Mode) | Incremental | Start Block | Load Data/ Start Profile |
| 1 | Block # | | | | | | | |
| 2 | Command Axis Number | | | Command Message Type | | | | |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Target Position Low Byte | | | | | | | |
| 5 | Target Position Low Middle Byte | | | | | | | |
| 6 | Target Position High Middle Byte | | | | | | | |
| 7 | Target Position High Byte | | | | | | | |

**Table 6-18.8 Command Message Type 02 hex Target Velocity - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Arm | Hard Stop | Smooth Stop | Direction (V. Mode) | Incremental | Start Block | Load Data/ Start Profile |
| 1 | Block # | | | | | | | |
| 2 | Command Axis Number | | | Command Message Type | | | | |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Target Velocity Low Byte | | | | | | | |
| 5 | Target Velocity Low Middle Byte | | | | | | | |
| 6 | Target Velocity High Middle Byte | | | | | | | |
| 7 | Target Velocity High Byte | | | | | | | |

**Table 6-18.9 Command Message Type 03 hex Acceleration - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Arm | Hard Stop | Smooth Stop | Direction (V. Mode) | Incremental | Start Block | Load Data/ Start Profile |
| 1 | Block # | | | | | | | |
| 2 | Command Axis Number | | | Command Message Type | | | | |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Acceleration Low Byte | | | | | | | |
| 5 | Acceleration Low Middle Byte | | | | | | | |
| 6 | Acceleration High Middle Byte | | | | | | | |
| 7 | Acceleration High Byte | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-18.10 Command Message Type 04 hex Deceleration - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Arm | Hard Stop | Smooth Stop | Direction (V. Mode) | Incremental | Start Block | Load Data/ Start Profile |
| 1 | Block # | | | | | | | |
| 2 | Command Axis Number | | | Command Message Type | | | | |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Deceleration Low Byte | | | | | | | |
| 5 | Deceleration Low Middle Byte | | | | | | | |
| 6 | Deceleration High Middle Byte | | | | | | | |
| 7 | Deceleration High Byte | | | | | | | |

**Table 6-18.11 Command Message Type 05 hex Torque - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Arm | Hard Stop | Smooth Stop | Direction (V. Mode) | Incremental | Start Block | Load Data/ Start Profile |
| 1 | Block # | | | | | | | |
| 2 | Command Axis Number | | | Command Message Type | | | | |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Torque Low Byte | | | | | | | |
| 5 | Torque Low Middle Byte | | | | | | | |
| 6 | Torque High Middle Byte | | | | | | | |
| 7 | Torque High Byte | | | | | | | |

**Table 6-18.12 Command Message Type 19 hex Motor Data Attribute - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Arm | Hard Stop | Smooth Stop | Direction (V. Mode) | Incremental | Start Block | Load Data/ Start Profile |
| 1 | Motor Data Attribute to Get | | | | | | | |
| 2 | Command Axis Number | | | Command Message Type | | | | |
| 3 | Motor Data Attribute to Set | | | | | | | |
| 4 | Motor Data Attribute Value Low Byte | | | | | | | |
| 5 | Motor Data Attribute Value Low Middle Byte | | | | | | | |
| 6 | Motor Data Attribute Value High Middle Byte | | | | | | | |
| 7 | Motor Data Attribute Value High Byte | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-18.13 Command Message Type 1A hex Position Controller Supervisor Attribute - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Arm | Hard Stop | Smooth Stop | Direction (V. Mode) | Incremental | Start Block | Load Data/ Start Profile |
| 1 | Position Controller Supervisor Attribute to Get | | | | | | | |
| 2 | Command Axis Number | | | Command Message Type | | | | |
| 3 | Position Controller Supervisor Attribute to Set | | | | | | | |
| 4 | Position Controller Supervisor Attribute Value Low Byte | | | | | | | |
| 5 | Position Controller Supervisor Attribute Value Low Middle Byte | | | | | | | |
| 6 | Position Controller Supervisor Attribute Value High Middle Byte | | | | | | | |
| 7 | Position Controller Supervisor Attribute Value High Byte | | | | | | | |

**Table 6-18.14 Command Message Type 1B hex Position Controller Attribute - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Arm | Hard Stop | Smooth Stop | Direction (V. Mode) | Incremental | Start Block | Load Data/ Start Profile |
| 1 | Position Controller Attribute to Get | | | | | | | |
| 2 | Command Axis Number | | | Command Message Type | | | | |
| 3 | Position Controller Attribute to Set | | | | | | | |
| 4 | Position Controller Attribute Value Low Byte | | | | | | | |
| 5 | Position Controller Attribute Value Low Middle Byte | | | | | | | |
| 6 | Position Controller Attribute Value High Middle Byte | | | | | | | |
| 7 | Position Controller Attribute Value High Byte | | | | | | | |

**Table 6-18.15 Command Message Type 1C hex Block Sequencer Attribute - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Arm | Hard Stop | Smooth Stop | Direction (V. Mode) | Incremental | Start Block | Load Data/ Start Profile |
| 1 | Block Sequencer Attribute to Get | | | | | | | |
| 2 | Command Axis Number | | | Command Message Type | | | | |
| 3 | Block Sequencer Attribute to Set | | | | | | | |
| 4 | Block Sequencer Attribute Value Low Byte | | | | | | | |
| 5 | Block Sequencer Attribute Value Low Middle Byte | | | | | | | |
| 6 | Block Sequencer Attribute Value High Middle Byte | | | | | | | |
| 7 | Block Sequencer Attribute Value High Byte | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-18.16 Command Message Type 1D hex Drive Data Attribute - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Arm | Hard Stop | Smooth Stop | Direction (V. Mode) | Incremental | Start Block | Load Data/ Start Profile |
| 1 | Drive Data Attribute to Get | | | | | | | |
| 2 | Command Axis Number | | | Command Message Type | | | | |
| 3 | Drive Data Attribute to Set | | | | | | | |
| 4 | Drive Data Attribute Value Low Byte | | | | | | | |
| 5 | Drive Data Attribute Value Low Middle Byte | | | | | | | |
| 6 | Drive Data Attribute Value High Middle Byte | | | | | | | |
| 7 | Drive Data Attribute Value High Byte | | | | | | | |

**Table 6-18.17 Command Message Type 1E hex Command Block Attribute - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Arm | Hard Stop | Smooth Stop | Direction (V. Mode) | Incremental | Start Block | Load Data/ Start Profile |
| 1 | Command Block Attribute to Get/Set | | | | | | | |
| 2 | Command Axis Number | | | Command Message Type | | | | |
| 3 | Command Block Instance to Get/Set | | | | | | | |
| 4 | Command Block Attribute Value Low Byte | | | | | | | |
| 5 | Command Block Attribute Value Low Middle Byte | | | | | | | |
| 6 | Command Block Attribute Value High Middle Byte | | | | | | | |
| 7 | Command Block Attribute Value High Byte | | | | | | | |

**Table 6-18.18 Command Message Type 1F hex Parameter - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Arm | Hard Stop | Smooth Stop | Direction (V. Mode) | Incremental | Start Block | Load Data/ Start Profile |
| 1 | Parameter Instance to Get | | | | | | | |
| 2 | Command Axis Number | | | Command Message Type | | | | |
| 3 | Parameter Instance to Set | | | | | | | |
| 4 | Parameter Value Low Byte | | | | | | | |
| 5 | Parameter Value Low Middle Byte | | | | | | | |
| 6 | Parameter Value High Middle Byte | | | | | | | |
| 7 | Parameter Value High Byte | | | | | | | |

## 6-18.5.2 Semantics

### 6-18.5.2.1 Load Data/ Start Profile

Set from zero to one to load command data. The transition of this bit from zero to one will also start a Profile Move when the command message type contained in the command message field is the message type that starts a Profile Move for the mode selected. Refer to Section 6-18.4.3 for an explanation of what commands start a Profile Move for a given mode.

### 6-18.5.2.2 Start Block

This bit is used to execute a Command block or Command block chain. Set from zero to one to execute a command block or command block chain.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 6-18.5.2.3 Incremental

This bit is used to define the position value as either absolute or incremental. 0 = absolute position value and 1 = incremental position value.

#### 6-18.5.2.4 Direction (V. Mode)

This bit is used to control the direction of the motor in Velocity mode. A 1 = forward, positive and a 0 = reverse, negative.

#### 6-18.5.2.5 Smooth Stop

This bit is used to bring the motor to a controlled stop at the currently implemented deceleration rate.

#### 6-18.5.2.6 Hard Stop

This bit is used to bring the motor to an immediate stop.

#### 6-18.5.2.7 Reg Arm

This bit is used to arm the registration input. When the registration input is triggered, the registration action will be executed.

#### 6-18.5.2.8 Enable

This bit is used to control the enable output. Clearing this bit will set the enable output inactive and the currently executing motion profile will be aborted.

#### 6-18.5.2.9 Block #

This byte defines the block number to be executed when the Start Block bit transitions from zero to one.

#### 6-18.5.2.10 Command Message Type

This field defines the Command Message Type

#### 6-18.5.2.11 Response Message Type

This field defines the Response Message Type

#### 6-18.5.2.12 Command Axis Number

These three bits define the Consumed Axis Connection attribute of the Position Controller Supervisor class. This attribute value specifies the instance number of all of the axis objects whose data are contained in the I/O command message. The command axis number is specified as shown in the table below:



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-18.19 Command Axis Numbers**

| Command Axis Number | Byte 2 | | |
|---|---|---|---|
| | Bit7 | Bit 6 | Bit 5 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 |

Note that axis 1 can be specified by either 0 or 1. Axis zero is not allowed.

#### 6-18.5.2.13 Target Position - Command Message Type 01 hex

This double word defines the Profile Move's Target Position in position units, when the Load Data /Start Profile bit transitions from zero to one.

#### 6-18.5.2.14 Target Velocity - Command Message Type 02 hex

This double word defines the Profile Move's Target Velocity in profile units, when the Load Data /Start Profile bit transitions from zero to one

#### 6-18.5.2.15 Acceleration - Command Message Type 03 hex

This double word defines the Profile Move's Acceleration in profile units, when the Load Data /Start Profile bit transitions from zero to one..

#### 6-18.5.2.16 Deceleration - Command Message Type 04 hex

This double word defines the Profile Move's Target Position in profile units, when the Load Data /Start Profile bit transitions from zero to one

#### 6-18.5.2.17 Torque - Command Message Type 05 hex

This double word is used to set the output torque, when the Load Data /Start Profile bit transitions from zero to one. The torque value will only take effect when in torque mode. (Position Controller Object Attribute 3 = 2)

#### 6-18.5.2.18 Attribute Value – Command Message Types 19 – 1E hex

This double word defines the value of the attribute to set, when the Load Data/Start Profile bit transitions from zero to one.

#### 6-18.5.2.19 Object Attribute to Get – Command Message Types 19 – 1D hex

This byte defines the object attribute to get the value of and return in the response message.

#### 6-18.5.2.20 Object Attribute to Set – Command Message Types 19 – 1D hex

This byte defines the object attribute to set to the new value defined by the Attribute Value when the Load Data/Start Profile bit transitions from zero to one.

#### 6-18.5.2.21 Command Block Attribute to Get/Set – Command Message Type 1E hex

This byte defines the attribute of the Command Block Object Instance to get/set, when the Load Data/Start Profile bit transitions from zero to one.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 6-18.5.2.22 Command Block Instance to Get/Set – Command Message Type 1E hex

This byte defines the instance of the Command Block Object for which the attribute is being get/set, when the Load Data/Start Profile bit transitions from zero to one.

#### 6-18.5.2.23 Parameter Instance to Get - Command Message Type 1F hex

This byte defines the instance of the parameter object to get the value of and return in the response message.

#### 6-18.5.2.24 Parameter Instance to Set - Command Message Type 1F hex

This byte defines the instance of the parameter object to set to the new value defined by the Parameter Value, when the Load Data/Start Profile bit transitions from zero to one.

#### 6-18.5.2.25 Parameter Value - Command Message Type 1F hex

This double word defines the value of the parameter to set, when the Load Data/Start Profile bit transitions from zero to one.

### 6-18.5.3 Response Message Types

The response message type is defined by byte 03 of the message. Bytes 00, 02 and 03 are the same for all response message types. Bytes 01 and 04 through 07 are defined by the Response Message Type code in byte 03.

**Table 6-18.20 Response Message Type 01 hex Actual Position - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Executing Block Number | | | | | | | |
| 2 | Load Complete | Block Fault | FE Fault | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Actual Position Low Byte | | | | | | | |
| 5 | Actual Position Low Middle Byte | | | | | | | |
| 6 | Actual Position High Middle Byte | | | | | | | |
| 7 | Actual Position High Byte | | | | | | | |

**Table 6-18.21 Response Message Type 02 hex Command Position - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Executing Block Number | | | | | | | |
| 2 | Load Complete | Block Fault | FE Fault | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Commanded Position Low Byte | | | | | | | |
| 5 | Commanded Position Low Middle Byte | | | | | | | |
| 6 | Commanded Position High Middle Byte | | | | | | | |
| 7 | Commanded Position High Byte | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-18.22 Response Message Type 03 hex Actual Velocity - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Executing Block Number | | | | | | | |
| 2 | Load Complete | Block Fault | FE Fault | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Actual Velocity Low Byte | | | | | | | |
| 5 | Actual Velocity Low Middle Byte | | | | | | | |
| 6 | Actual Velocity High Middle Byte | | | | | | | |
| 7 | Actual Velocity High Byte | | | | | | | |

**Table 6-18.23 Response Message Type 04 hex Command Velocity - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Executing Block Number | | | | | | | |
| 2 | Load Complete | Block Fault | FE Fault | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Commanded Velocity Low Byte | | | | | | | |
| 5 | Commanded Velocity Low Middle Byte | | | | | | | |
| 6 | Commanded Velocity High Middle Byte | | | | | | | |
| 7 | Commanded Velocity High Byte | | | | | | | |

**Table 6-18.24 Response Message Type 05 hex Torque - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Executing Block Number | | | | | | | |
| 2 | Load Complete | Block Fault | FE Fault | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Torque Low Byte | | | | | | | |
| 5 | Torque Low Middle Byte | | | | | | | |
| 6 | Torque High Middle Byte | | | | | | | |
| 7 | Torque High Byte | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-18.25 Response Message Type 06 hex Captured Home Position - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Executing Block Number | | | | | | | |
| 2 | Load Complete | Block Fault | FE Fault | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Home Position Low Byte | | | | | | | |
| 5 | Home Position Low Middle Byte | | | | | | | |
| 6 | Home Position High Middle Byte | | | | | | | |
| 7 | Home Position High Byte | | | | | | | |

**Table 6-18.26 Response Message Type 07 hex Captured Index Position - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Executing Block Number | | | | | | | |
| 2 | Load Complete | Block Fault | FE Fault | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Index Position Low Byte | | | | | | | |
| 5 | Index Position Low Middle Byte | | | | | | | |
| 6 | Index Position High Middle Byte | | | | | | | |
| 7 | Index Position High Byte | | | | | | | |

**Table 6-18.27 Response Message Type 08 hex Captured Registration Position - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Executing Block Number | | | | | | | |
| 2 | Load Complete | Block Fault | FE FAULT | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Registration Position Low Byte | | | | | | | |
| 5 | Registration Position Low Middle Byte | | | | | | | |
| 6 | Registration Position High Middle Byte | | | | | | | |
| 7 | Registration Position High Byte | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-18.28 Response Message Type 14 hex Command/Response Error – Required**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Reserved = 0 | | | | | | | |
| 2 | Load Complete | Block Fault | FE Fault | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | General Error Code | | | | | | | |
| 5 | Additional Code | | | | | | | |
| 6 | Copy of Command Message Byte 2 | | | | | | | |
| 7 | Copy of Command Message Byte 3 | | | | | | | |

**Table 6-18.29 Response Message Type 19 hex Motor Data Attribute - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Motor Data Attribute to Get | | | | | | | |
| 2 | Load Complete | Block Fault | FE Fault | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Motor Data Attribute Value Low Byte | | | | | | | |
| 5 | Motor Data Attribute Value Low Middle Byte | | | | | | | |
| 6 | Motor Data Attribute Value High Middle Byte | | | | | | | |
| 7 | Motor Data Attribute Value High Byte | | | | | | | |

**Table 6-18.30 Response Message Type 1A hex Position Controller Supervisor Attribute - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Position Controller Supervisor Attribute to Get | | | | | | | |
| 2 | Load Complete | Block Fault | FE Fault | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Position Controller Supervisor Attribute Value Low Byte | | | | | | | |
| 5 | Position Controller Supervisor Attribute Value Low Middle Byte | | | | | | | |
| 6 | Position Controller Supervisor Attribute Value High Middle Byte | | | | | | | |
| 7 | Position Controller Supervisor Attribute Value High Byte | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-18.31 Response Message Type 1B hex Position Controller Attribute - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Position Controller Attribute to Get | | | | | | | |
| 2 | Load Complete | Block Fault | FE Fault | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Position Controller Attribute Value Low Byte | | | | | | | |
| 5 | Position Controller Attribute Value Low Middle Byte | | | | | | | |
| 6 | Position Controller Attribute Value High Middle Byte | | | | | | | |
| 7 | Position Controller Attribute Value High Byte | | | | | | | |

**Table 6-18.32 Response Message Type 1C hex Block Sequencer Attribute - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Block Sequencer Attribute to Get | | | | | | | |
| 2 | Load Complete | Block Fault | FE Fault | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Block Sequencer Attribute Value Low Byte | | | | | | | |
| 5 | Block Sequencer Attribute Value Low Middle Byte | | | | | | | |
| 6 | Block Sequencer Attribute Value High Middle Byte | | | | | | | |
| 7 | Block Sequencer Attribute Value High Byte | | | | | | | |

**Table 6-18.33 Response Message Type 1D hex Drive Data Attribute - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Drive Attribute to Get | | | | | | | |
| 2 | Load Complete | Block Fault | FE Fault | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Drive Attribute Value Low Byte | | | | | | | |
| 5 | Drive Attribute Value Low Middle Byte | | | | | | | |
| 6 | Drive Attribute Value High Middle Byte | | | | | | | |
| 7 | Drive Attribute Value High Byte | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-18.34 Response Message Type 1E hex Command Block Attribute - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Command Block Attribute to Get | | | | | | | |
| 2 | Load Complete | Block Fault | FE Fault | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Command Block Attribute Value Low Byte | | | | | | | |
| 5 | Command Block Attribute Value Low Middle Byte | | | | | | | |
| 6 | Command Block Attribute Value High Middle Byte | | | | | | | |
| 7 | Command Block Attribute Value High Byte | | | | | | | |

**Table 6-18.35 Response Message Type 1F hex Parameter - Optional**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Enable | Reg Level | Home Level | Current Direction | General Fault | On Target Position | Block in execution | Profile in Progress |
| 1 | Parameter Instance to Get | | | | | | | |
| 2 | Load Complete | Block Fault | FE Fault | Negative Limit | Positive Limit | Rev Limit | Fwd Limit | Fault Input Fault |
| 3 | Response Axis Number | | | Response Message Type | | | | |
| 4 | Parameter Value Low Byte | | | | | | | |
| 5 | Parameter Value Low Middle Byte | | | | | | | |
| 6 | Parameter Value High Middle Byte | | | | | | | |
| 7 | Parameter Value High Byte | | | | | | | |

### 6-18.5.4 Semantics

#### 6-18.5.4.1 Profile in Progress

This bit indicates that a profile move is in progress.

#### 6-18.5.4.2 Block in Execution

This bit indicates that a block is in execution. The command block that is currently being executed is returned in byte 1.

#### 6-18.5.4.3 On Target Position

This bit indicates whether or not the motor is on the last targeted position. (1 = Current position equals the last target position.)

#### 6-18.5.4.4 General Fault

This bit indicates the logical “or” of all fault conditions.

#### 6-18.5.4.5 Current Direction

This bit shows the current direction of the motor. If the motor is not moving the bit will indicate the direction of the last commanded move. 0 = reverse or negative direction and 1 = forward or positive direction.

#### 6-18.5.4.6 Home Level

This bit reflects the level of the home input.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 6-18.5.4.7 Reg Level

This bit reflects the level of the registration input.

#### 6-18.5.4.8 Enable

This bit indicates the state of the enable output. A 1 indicates the enable output is active.

#### 6-18.5.4.9 Executing Block #

This byte defines the currently executing block if the Block In Execution bit is active.

#### 6-18.5.4.10 Fault Input Fault

This bit indicates that the fault input is active.

#### 6-18.5.4.11 Fwd Limit

This bit indicates that the forward input is active.

#### 6-18.5.4.12 Rev Limit

This bit indicates that the reverse input is active.

#### 6-18.5.4.13 Positive Limit

This bit indicates that the motor has attempted to travel past the programmed positive limit position. This bit remains valid until the motor is moved within the limits or the programmed limit value is set greater than the current position.

#### 6-18.5.4.14 Negative Limit

This bit indicates that the motor has attempted to travel past the programmed negative limit position. This bit remains valid until the motor is moved within the limits or the programmed limit value is set less than the current position.

#### 6-18.5.4.15 FE Fault

This bit indicates that a following error fault has occurred. This fault occurs when the following error, or difference between the commanded and actual position, exceeds the programmed allowable following error.

#### 6-18.5.4.16 Block Fault

This bit indicates that a block execution fault has occurred. When this happens block execution and motion will cease. This bit is reset when Block Sequencer Block Fault Code attribute (Block Sequencer class, attribute 5) is read.

#### 6-18.5.4.17 Load Complete

This bit indicates that the command data contained in the command message has been successfully loaded into the device**.**

#### 6-18.5.4.18 Response Message Type

This byte defines the Response Message Type

#### 6-18.5.4.19 Response Axis Number

These three bits report the Produced Axis Connection attribute of the Position Controller Supervisor class. This attribute value specifies the instance number of all of the axis objects whose data is contained in the I/O response message.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-18.36Table 6-18.37 Response Axis Numbers**

| Response Axis Number | Byte 3 | | |
|---|---|---|---|
| | Bit7 | Bit 6 | Bit 5 |
| 1 | 0 | 0 | 0 |
| | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 |

Note that axis 1 can be reported as either binary 0 or 1. Axis zero is not allowed.

### 6-18.5.4.20 Actual Position - Response Message Type 01 hex

This double word reflects the actual position in position units. If position feedback is not used, this word will report the commanded position.

### 6-18.5.4.21 Commanded Position - Response Message Type 02 hex

This double word reflects the commanded or calculated position in position units.

### 6-18.5.4.22 Actual Velocity - Response Message Type 03 hex

This double word reflects the actual velocity in profile units.

### 6-18.5.4.23 Command Velocity - Response Message Type 04 hex

This double word reflects the commanded or calculated velocity in profile units.

### 6-18.5.4.24 Torque - Response Message Type 05 hex

This double word reflects the torque.

### 6-18.5.4.25 Home Position - Response Message Type 06 hex

This double word reflects the captured home position in position units.

### 6-18.5.4.26 Index Position - Response Message Type 07 hex

This double word reflects the captured index position in position units.

### 6-18.5.4.27 Registration Position - Response Message Type 08 hex

This double word reflects the captured registration position in position units.

### 6-18.5.4.28 General Error Code – Response Message Type 14 hex

This byte identifies an error has been encountered. The table below summarizes specific behavior for the Position Controller Profile. See Appendix B for a complete list of General Error codes.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-18.38 Additional Error Codes for Position Controller Profile**

| General Error Code | Additional Code | Error Description | Semantics |
|---|---|---|---|
| $08_{hex}$ | $01_{hex}$ | Service Not Supported | Command Message type not supported. Additional code 01 takes precedence over additional code 02. [1] |
| | $02_{hex}$ | Service Not Supported | Response message type not supported. |
| $05_{hex}$ | $01_{hex}$ | Path Destination Unknown | Consumed axis number was requested that does not exist in the drive. |
| | $02_{hex}$ | Path Destination Unknown | A produced axis number was requested that does not exist in the drive. |
| $09_{hex}$ | $FF_{hex}$ | Invalid Attribute Value | Load value is out of range. |
| $0E_{hex}$ | $FF_{hex}$ | Attribute not Settable | A request to modify a non-modifiable attribute was received. |
| $13_{hex}$ | $FF_{hex}$ | Not Enough Data | I/O command message contained fewer than 8 bytes. |
| $14_{hex}$ | $FF_{hex}$ | Attribute Not Supported | Attribute specified in request was not supported. |

Table Footnotes

1 If Response Message Type is supported and Command Message Type is not supported, a General Error Code 08, Additional Code 01 shall be returned.

### 6-18.5.4.29 Additional Code – Response Message Type 14 hex

This byte contains an object/service-specific value that further describes the error condition. If the responding object has no additional information to specify, then the value $FF_{hex}$ is placed within this field.

### 6-18.5.4.30 Attribute Value – Response Message Types 19 – 1E hex

This double word reflects the value of the attribute to get.

### 6-18.5.4.31 Object Attribute to Get – Response Message Types 19 – 1E hex

This byte defines the object attribute from which to get the value.

### 6-18.5.4.32 Parameter Instance to Get - Response Message Type 1F hex

This byte defines the instance of the parameter object to get the value of and return in the response message.

### 6-18.5.4.33 Parameter Value - Response Message Type 1F hex

This double word reflects the value of the parameter to get.

## 6-18.6 Mapping I/O Message Data Attribute Components

The following table indicates the I/O Data Attribute mapping for the Position Controller Profile Command Messages.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-18.39 I/O Assembly Data Mapping – Command Messages**

| Data Component | Class | | Instance Number | Attribute | | Data |
|---|---|---|---|---|---|---|
| Name | Name | # | | Name | # | Type |
| Load Data/ Start Profile | Position Controller | $25_{hex}$ | 1-7 | Load Data/ Profile Handshake | 11 | BOOL |
| Start Block | Block Sequencer | $26_{hex}$ | 1-7 | Block Execute | 2 | BOOL |
| Incremental | Position Controller | $25_{hex}$ | 1-7 | Incremental | 10 | BOOL |
| Direction | Position Controller | $25_{hex}$ | 1-7 | Direction | 23 | BOOL |
| Smooth Stop | Position Controller | $25_{hex}$ | 1-7 | Hard Stop | 20 | BOOL |
| Hard Stop | Position Controller | $25_{hex}$ | 1-7 | Hard Stop | 21 | BOOL |
| Registration Arm | Position Ctrl Supervisor | $24_{hex}$ | 1-7 | Arm Registration | 21 | BOOL |
| Enable | Position Controller | $25_{hex}$ | 1-7 | Enable | 17 | BOOL |
| Block # | Block Sequencer | $26_{hex}$ | 1-7 | Block | 1 | USINT |
| Command Axis Number | Position Ctrl Supervisor Class | $24_{hex}$ | 0 | Consumed Axis Number | 32 | USINT |
| Command Message Type | Position Ctrl Supervisor | $24_{hex}$ | 1-7 | Cmd Message Type | 6 | USINT |
| Response Axis Number | Position Ctrl Supervisor Class | $24_{hex}$ | 1-7 | Produced Axis Number | 33 | USINT |
| Response Message Type | Position Ctrl Supervisor | $24_{hex}$ | 1-7 | Rspnc Message Type | 7 | USINT |
| Target Position | Position Controller | $25_{hex}$ | 1-7 | Target Position | 6 | DINT |
| Target Velocity | Position Controller | $25_{hex}$ | 1-7 | Target velocity | 7 | DINT |
| Acceleration | Position Controller | $25_{hex}$ | 1-7 | Acceleration | 8 | DINT |
| Deceleration | Position Controller | $25_{hex}$ | 1-7 | Deceleration | 9 | DINT |
| Torque | Position Controller | $25_{hex}$ | 1-7 | Torque | 25 | DINT |
| Parameter Value | Parameter | $0F_{hex}$ | 1-255 | Parameter Value | 1 | Determined by instance of parameter |

The following table indicates the I/O Data Attribute mapping for the Position Controller Profile Response Messages.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Position Controller Device, Type: $10_{Hex}$

**Table 6-18.40 I/O Assembly Data Mapping – Response Messages**

| Data Component Name | Class | | Instance | Attribute | | Data |
|---|---|---|---|---|---|---|
| | Name | # | Number | Name | # | Type |
| Profile in Progress | Position Controller | $25_{hex}$ | 1-7 | Profile in Progress | 11 | BOOL |
| Block in Execution | Block Sequencer | $26_{hex}$ | 1-7 | Block Execute | 2 | BOOL |
| On Target Position | Position Controller | $25_{hex}$ | 1-7 | On Position | 12 | BOOL |
| General Fault | Position Ctrl Supervisor | $24_{hex}$ | 1-7 | General Fault | 5 | BOOL |
| Direction | Position Controller | $25_{hex}$ | 1-7 | Direction | 23 | BOOL |
| Home Level | Position Ctrl Supervisor | $24_{hex}$ | 1-7 | Home Input Level | 16 | BOOL |
| Reg Level | Position Ctrl Supervisor | $24_{hex}$ | 1-7 | Registration Input Level | 22 | BOOL |
| Enable State | Position Controller | $25_{hex}$ | 1-7 | Enable | 17 | BOOL |
| Executing Block # | Block Sequencer | $26_{hex}$ | 1-7 | Current Block | 3 | USINT |
| Fault Input Fault | Position Ctrl Supervisor | $24_{hex}$ | 1-7 | Fault Input | 8 | BOOL |
| Fwd Limit | Position Controller | $25_{hex}$ | 1-7 | Fwd Limit | 50 | BOOL |
| Rev Limit | Position Controller | $25_{hex}$ | 1-7 | Rev Limit | 51 | BOOL |
| Positive Limit | Position Controller | $25_{hex}$ | 1-7 | Positive Limit Triggered | 56 | BOOL |
| Negative Limit | Position Controller | $25_{hex}$ | 1-7 | Negative Limit Triggered | 57 | BOOL |
| FE Fault | Position Controller | $25_{hex}$ | 1-7 | Following Error Fault | 47 | BOOL |
| Block Fault | Block Sequencer | $26_{hex}$ | 1-7 | Block Fault | 4 | BOOL |
| Load Complete | Position Controller | $25_{hex}$ | 1-7 | Load Data Complete | 58 | BOOL |
| Response Axis Number | Position Ctrl Supervisor Class | $24_{hex}$ | 0 | Produced Axis Number | 33 | USINT |
| Response Message Type | Position Ctrl Supervisor | $24_{hex}$ | 1-7 | Rspnc Message Type | 7 | USINT |
| Actual Position | Position Controller | $25_{hex}$ | 1-7 | Actual Position | 13 | DINT |
| Commanded Position | Position Controller | $25_{hex}$ | 1-7 | Commanded Position | 15 | DINT |
| Actual Velocity | Position Controller | $25_{hex}$ | 1-7 | Actual Velocity | 14 | DINT |
| Commanded Velocity | Position Controller | $25_{hex}$ | 1-7 | Commanded Velocity | 16 | DINT |
| Torque | Position Controller | $25_{hex}$ | 1-7 | Torque | 25 | DINT |
| Home Position | Position Ctrl Supervisor | $24_{hex}$ | 1-7 | Home Position | 17 | DINT |
| Index Position | Position Ctrl Supervisor | $24_{hex}$ | 1-7 | Index Position | 18 | DINT |
| Registration Position | Position Ctrl Supervisor | $24_{hex}$ | 1-7 | Registration Position | 24 | DINT |
| Parameter Value | Parameter | $0F_{hex}$ | 1-255 | Parameter Value | 1 | Determined by instance of parameter |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-19 Motor Overload Device

**Device Type: 03 Hex**

The Motor Overload device profile is part of a "Hierarchy of Motor Control Devices" that are supported by CIP. This hierarchy includes:

- Contactors, Overloads, and Across the Line Motor Starters
- Softstarters
- AC/DC Drives
- Servo Drives

Devices within this hierarchy use a common Control Supervisor object to control state behavior of the device. Devices within this hierarchy also support a hierarchy of "IO Assembly Instance" definitions which are used to pass control and status information to and from a device. Assembly instances are numbered so that each device type is assigned a range of instance numbers, with higher functionality devices supporting higher instance numbers. Devices within the hierarchy can choose to support some instance numbers that are lower than theirs in the hierarchy. For example, an AC Drive may choose to support some instances that are defined for Across the Line Motor Starters. This makes it easier to interchange drives and starters within a system.

This profile makes Motor Starters of the same device type inter-operable, but not directly interchangeable without doing configuration through a unit's local interface, a network configuration tool or other means of configuring outside the CIP interface.

## 6-19.1 Object Model

The Object Model in Figure 6-19.1 represents a Motor Overload. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-19.1 Objects Present in a Motor Overload Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| Message Router | Optional | 1 |
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Optional | 1 |
| Parameter | Optional | - |
| Control Supervisor | Required | 1 |
| Overload | Required | - |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-19.1 Object Model for a Motor Overload Device**

## 6-19.2 How Objects Affect Behavior

The objects for this device affect the device's behavior as shown in the table below.

**Table 6-19.2 Object Effect on Behavior**

| Object | Effect on behavior |
|---|---|
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines I/O data format |
| Parameter | Provides a public interface to device configuration data |
| Control Supervisor | Manages motor functions and operational states |
| Overload | Implements overload |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-19.3 Defining Object Interfaces

The objects in the Motor Overload have the interfaces listed in the following table:

**Table 6-19.3 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| Parameter | Message Router |
| Control Supervisor | Message Router, Assembly or Parameter Object |
| Overload | Message Router, Assembly or Parameter Object |

## 6-19.4 Contactor Interface and Behavior

**Figure 6-19.2 Behavior Diagram**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-19.5 I/O Assembly Instances

The IO Assembly Instance definitions in this section define the format of the "data" attribute (attribute 3) for IO Assembly Instances. Through the use of predefined instance definitions, IO Assemblies support a hierarchy of motor control devices. The device hierarchy includes motor starters, soft starters, AC and DC drives, and servo drives. Assembly Instances are numbered within the hierarchy so that each device type is assigned a range of Assembly Instance numbers, with higher functionality devices supporting higher instance numbers. **Devices in the hierarchy can choose to support instance numbers that are lower than theirs in the hierarchy.** For example a Softstart may choose to support some IO Assemblies that are defined for Overload. The following table shows the Assembly Instance numbering for the motor control device hierarchy.

**Table 6-19.4 I/O Assembly Instances – Motor Control Device Hierarchy**

| Profile | I/O Type | Instance Range |
|---|---|---|
| Contactors, Overloads and | Output | 1-19 |
| Starters | Input | 50-69 |
| AC/DC Drive | Output | 20-29 |
| | Input | 70-79 |
| Servo Drive | Output | 30-49 |
| | Input | 80-99 |

The following IO Assembly Instances are defined for Overloads.

**Table 6-19.5 I/O Assembly Instances - Overloads**

| Instance | Type | Name |
|---|---|---|
| 2 | Output | Basic Overload |
| 50 | Input | Basic Overload |
| 51 | Input | Extended Overload |

If a bit is not used in an IO Assembly, it is reserved for use in other Assemblies. Reserved bits in Output Assemblies are ignored by the consuming device. Reserved bits in Input Assemblies are set to zero by the producing device.

### 6-19.5.1 Connection Paths to I/O Assembly Instances

The IO Assembly Instances are chosen for IO Connections by setting the "produced_connection_path" (attribute 14) and "consumed_connection_path" (attribute 16) attributes in the appropriate connection object.

Motor Control Devices use the Symbolic Segment Type (see Appendix C) to specify paths to the IO Assembly Instances in the Motor Control Hierarchy. IO Assembly Instances are represented by ASCII strings that contain the hex number of the Assembly Instance whose path is to be chosen.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The following example shows the Symbolic Segment used to specify Output Assembly Instance 20 (14 hex).

**Figure 6-19.3 Symbolic Segment Example**

| Segment Type | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| Segment type (symbolic) | | | Symbol size in bytes (2 bytes) | | | | |

| First Character | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| ASCII 1 (31 hex) | | | | | | | |

| Second Character | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| ASCII 4 (34 hex) | | | | | | | |

## 6-19.6 I/O Assembly Data Attribute Format

### 6-19.6.1 Output Assembly Data Attribute Format

**Table 6-19.6 I/O Assembly Instance 2: Basic Overload**

| This is the only required output assembly for the device type Motor Overload (03hex) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | Reserved | Reserved | Reserved | Reserved | Reserved | FaultReset | Reserved | Reserved |

### 6-19.6.2 Input Assembly Data Attribute Format

**Table 6-19.7 I/O Assembly Instance 50: Basic Overload**

| This is the only required input assembly for the device type Motor Overload.(03hex). | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Faulted/ Trip |

**Table 6-19.8 I/O Assembly Instance 51: Extended Overload**

| This assembly uses some optional attributes | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Warning | Faulted/ Trip |

## 6-19.7 Mapping I/O Assembly Data Attribute Components

### 6-19.7.1 Mapping for Output Assembly Data Components

**Table 6-19.9 Output Assembly Data Mapping**

| Data Name | Class Name | Class Number | Instance | Attribute Name | Attribute Number |
|---|---|---|---|---|---|
| Fault Reset | Control Supervisor | 29 | 1 | FaultRst | 12 |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-19.7.2 Mapping for Input Assembly Data Components

**Table 6-19.10 Input Assembly Data Mapping**

| Data Name | Class Name | Class Number | Instance | Attribute Name | Attribute Number |
|---|---|---|---|---|---|
| Faulted/ Trip | Control Supervisor | 0x29 | 1 | Faulted | 10 |
| Warning | Control Supervisor | 0x29 | 1 | Warning | 11 |

## 6-19.8 Defining Device Configuration

Public access to the Control Supervisor Object and the Overload Object must be supported for configuration of a Motor Overload device. If supported, optional Parameter Objects may be used to access the various configuration attributes in the Control Supervisor Object and the Overload Object.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-20 Weigh Scale Device

The profile for this device type will be defined by the ODVA.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-21 Encoder Device

**Device Type: 22 Hex**

## 6-21.1 Introduction

Encoders are used to detect positions of any kind of machines. These devices could be used for following applications: Sensing of angles, distances, tracks, velocity and motion control. This profile covers the measuring principle of absolute and incremental systems as well as the mechanical specification of rotary and linear devices.

## 6-21.2 Object Model

The Object Model in Figure 6-21.1 represents an encoder. The table below indicates:

- The object class
- Whether or not the class is required
- The number of instances present in each class

**Table 6-21.1 Objects Present in an Encoder Device**

| Object Class | Optional / Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Required | at least 1 I/O input assembly |
| Parameter | Optional | - |
| Position Sensor | Required | 1 |

**Figure 6-21.1 Object Model for an Encoder Device**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-21.3 How Objects Affect Behavior

Objects supported for the encoder affect the device's behavior as shown in the table below.

**Table 6-21.2 Object Effect on Behavior**

| Object | Effect on Behavior |
|---|---|
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines I/O data format |
| Parameter | Provides a public interface to the device's configuration data |
| Position Sensor | Affects Value (attribute) |

## 6-21.4 Defining Object Interfaces

Objects supported for the encoder have the interfaces listed in the table below.

**Table 6-21.3 Objects Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| Parameter | Message Router |
| Position Sensor | Message Router, Assembly Object or Parameter Object |

## 6-21.5 I/O Assembly Instances

The following table identifies the I/O Assembly instances, which should be supported by the encoder device.

**Table 6-21.4 I/O Assembly Instances**

| Number | Required/Optional | Type | Name |
|---|---|---|---|
| 1 | Required | Input | Position Value 1 Axis |
| 2 | Optional | Input | Position Value 1 axis & Warning Flag 1 axis & Alarm Flag 1 axis |
| 3 | Optional | Input | Position Value 1 axis & Velocity 1 axis |
| 4 | Optional | Input | Position Value 1 axis & Position Value 2 axis |
| 5 | Optional | Input | Position Value 1 axis & Velocity 1 axis<br>&<br>Position Value 2 axis & Velocity 2 axis |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-21.6 I/O Assembly Data Attribute Format

The I/O assembly data attributes have the format shown below.

**Table 6-21.5 I/O Assembly Data Attribute Format**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | Position Value 1 axis (low byte) | | | | | | | |
| | 1 | Position Value 1 axis | | | | | | | |
| | 2 | Position Value 1 axis | | | | | | | |
| | 3 | Position Value 1 axis (high byte) | | | | | | | |
| 2 | 0 | Position Value 1 axis (low byte) | | | | | | | |
| | 1 | Position Value 1 axis | | | | | | | |
| | 2 | Position Value 1 axis | | | | | | | |
| | 3 | Position Value 1 axis (high byte) | | | | | | | |
| | 4 | Vendor Specific | | | Reserved by CIP | | | Warning Flag 1 Axis | Alarm Flag 1 Axis |
| 3 | 0 | Position Value 1 axis (low byte) | | | | | | | |
| | 1 | Position Value 1 axis | | | | | | | |
| | 2 | Position Value 1 axis | | | | | | | |
| | 3 | Position Value 1 axis (high byte) | | | | | | | |
| | 4 | Velocity 1 axis (low byte) | | | | | | | |
| | 5 | Velocity 1 axis | | | | | | | |
| | 6 | Velocity 1 axis | | | | | | | |
| | 7 | Velocity 1 axis (high byte) | | | | | | | |
| 4 | 0 | Position Value 1 axis (low byte) | | | | | | | |
| | 1 | Position Value 1 axis | | | | | | | |
| | 2 | Position Value 1 axis | | | | | | | |
| | 3 | Position Value 1 axis (high byte) | | | | | | | |
| | 4 | Position Value 2 axis (low byte) | | | | | | | |
| | 5 | Position Value 2 axis | | | | | | | |
| | 6 | Position Value 2 axis | | | | | | | |
| | 7 | Position Value 2 axis (high byte) | | | | | | | |
| 5 | 0 | Position Value 1 axis (low byte) | | | | | | | |
| | 1 | Position Value 1 axis | | | | | | | |
| | 2 | Position Value 1 axis | | | | | | | |
| | 3 | Position Value 1 axis (high byte) | | | | | | | |
| | 4 | Velocity 1 axis (low byte) | | | | | | | |
| | 5 | Velocity 1 axis | | | | | | | |
| | 6 | Velocity 1 axis | | | | | | | |
| | 7 | Velocity 1 axis (high byte) | | | | | | | |
| | 8 | Position Value 2 axis (low byte) | | | | | | | |
| | 9 | Position Value 2 axis | | | | | | | |
| | 10 | Position Value 2 axis | | | | | | | |
| | 11 | Position Value 2 axis (high byte) | | | | | | | |
| | 12 | Velocity 2 axis (low byte) | | | | | | | |
| | 13 | Velocity 2 axis | | | | | | | |
| | 14 | Velocity 2 axis | | | | | | | |
| | 15 | Velocity 2 axis (high byte) | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-21.7 Mapping I/O Assembly Data Attribute Components

The following table indicates the I/O Assembly Data Attribute mapping for the Encoder Profile.

**Table 6-21.6 I/O Assembly Data Mapping**

| Data Component | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Position Value 1 axis | Position Sensor | $23_{hex}$ | 1 | Position Value | 10 |
| Velocity 1 axis | Position Sensor | $23_{hex}$ | 1 | Velocity | 24 |
| Warning Flag 1 axis | Position Sensor | $23_{hex}$ | 1 | Warning Flag | 49 |
| Alarm Flag 1 axis | Position Sensor | $23_{hex}$ | 1 | Alarm Flag | 46 |
| Position Value 2 axis | Position Sensor | $23_{hex}$ | 2 | Position Value | 10 |
| Velocity 2 axis | Position Sensor | $23_{hex}$ | 2 | Velocity | 24 |

## 6-21.8 Defining Device Configuration

Public access to the Position Sensor Object must be supported for configuration of an encoder device. If supported, the optional Parameter Objects may be used to access the various configuration attributes in the Position Sensor Object.

Encoder devices may contain (but are not limited to) any of the Parameter Object instances listed in the table below. Suggested parameter names are also given in the table. The set of parameters instances that are supported by a drive should be numbered sequentially with lower instance numbers assigned to parameters that appear earlier in the table. Vendor specific parameter instances should be numbered sequentially following the instances that appear in the following table.

Parameter Object instances may be implemented as EDS file definitions, parameter stubs, or full parameter objects. See Chapter 5 for a definition of the Parameter Object and an explanation of how it is used for configuration.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-21.9 Mapping Parameter Object Data

The following table indicates the Parameter Object data mapping for encoder devices.

**Table 6-21.7 Parameter Object Data Mapping**

| Configuration Parameter | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Direct Counting Toggle | Position Sensor | $23_{hex}$ | 1 | Direction Counting Toggle | 12 |
| Commissioning Diagnostic Control | Position Sensor | $23_{hex}$ | 1 | Commissioning Diagnostic Control | 12 |
| Scaling Function Control | Position Sensor | $23_{hex}$ | 1 | Scaling Function Control | 14 |
| Position Format | Position Sensor | $23_{hex}$ | 1 | Position Format | 15 |
| Measuring units per Span | Position Sensor | $23_{hex}$ | 1 | Meausring Units per Span | 16 |
| Total Measuring Range in measuring units | Position Sensor | $23_{hex}$ | 1 | Total Measuring Range in measuring units | 17 |
| Position Measuring Increment | Position Sensor | $23_{hex}$ | 1 | Position Measuring Increment | 18 |
| Preset Value | Position Sensor | $23_{hex}$ | 1 | Preset Value | 19 |
| Position Value 1 axis | Position Sensor | $23_{hex}$ | 1 | Position Value | 10 |
| Position Value 2 axis | Position Sensor | $23_{hex}$ | 2 | Position Value | 10 |
| Operating Status | Position Sensor | $23_{hex}$ | 1 | Operating Status | 41 |
| Physical Resolution Span | Position Sensor | $23_{hex}$ | 1 | Physical resolution span | 42 |
| Number of Span | Position Sensor | $23_{hex}$ | 1 | Number of Span | 43 |
| Alarm Flag 1 axis | Position Sensor | $23_{hex}$ | 1 | Alarm Flag | 46 |
| Alarm Flag 2 axis | Position Sensor | $23_{hex}$ | 2 | Alarm Flag | 46 |
| Alarms | Position Sensor | $23_{hex}$ | 1 | Alarms | 44 |
| Supported Alarms | Position Sensor | $23_{hex}$ | 1 | Supported Alarms | 45 |
| Warning Flag 1 axis | Position Sensor | $23_{hex}$ | 1 | Warning Flag | 49 |
| Warning Flag 2 axis | Position Sensor | $23_{hex}$ | 2 | Warning Flag | 49 |
| Warnings | Position Sensor | $23_{hex}$ | 1 | Warnings | 47 |
| Supported Warnings | Position Sensor | $23_{hex}$ | 1 | Supported Warnings | 48 |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-22 Resolver Device

## Device Type: 09hex

A Resolver mechanically or otherwise detects the absolute position of a shaft. The position information is represented as a binary integer value.

## 6-22.1 Object Model

The Object Model in Figure 6-22.1 represents a resolver. The table below indicates

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

Chapter 5, The CIP Object Library, provides more details about these objects.

**Table 6-22.1 Objects Present in a Resolver Device**

| Object Class | Optional / Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Required | at least 1 I/O input assembly |
| Parameter | Optional | 4 |
| Position Sensor | Required | 1 |

**Figure 6-22.1 Object Model for a Resolver**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-22.2 How Objects Affect Behavior

The objects in this device affect the device's behavior as shown in the following table.

**Table 6-22.2 Object Effect on Behavior**

| Object | Effect on Behavior |
|---|---|
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines I/O and/or configuration data format |
| Parameter | Provides a public interface to the device's configuration data |
| Position Sensor | Affects Value (attribute), Cam (attribute) |

## 6-22.3 Defining Object Interfaces

The objects in this device have the interfaces listed in the following table.

**Table 6-22.3 Objects Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| Parameter | Message Router |
| Position Sensor | Message Router, Assembly Object or Parameter Object |

## 6-22.4 I/O Assembly Instances

The following table identifies the I/O assembly instance supported by the Resolver device.

**Table 6-22.4 I/O Assembly Instances**

| Number | Required/Optional | Type | Name |
|---|---|---|---|
| 1 | Optional* | Input | Value |
| 2 | Optional* | Input | Value/Cam |
| 3 | Optional | Output | SetZero |
| *At least 1 input assembly is required | | | |

## 6-22.5 I/O Assembly Data Attribute Format

The I/O Assembly Data Attributes have the format shown below.

**Table 6-22.5 I/O Assembly Data Attribute Format**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | | | | | | | | |
| | 1 | Value | | | | | | | |
| | 2 | | | | | | | | |
| | 3 | | | | | | | | |
| 2 | 0 | | | | | | | | |
| | 1 | Value | | | | | | | |
| | 2 | | | | | | | | |
| | 3 | | | | | | | | |
| | 4 | Reserved (zero) | | | | | | | CAM |
| 3 | 0 | Reserved (zero) | | | | | | | SetZero |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-22.6 Mapping I/O Assembly Data Attribute Components

The following table indicates the I/O Assembly Data Attribute mapping for the Resolver device.

**Table 6-22.6 I/O Assembly Data Mapping**

| Data Component | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Value | Position Sensor | 23hex | 1 | Value | 3 |
| CAM | Position Sensor | 23hex | 1 | CAM | 4 |
| SetZero | Position Sensor | 23hex | 1 | SetZero | 9 |

## 6-22.7 Configuration Assembly Instances

The following table identifies the configuration assembly instance supported by the Resolver device.

**Table 6-22.7 Configuration Assembly Instances**

| Number | Required/Optional | Name |
|---|---|---|
| 40 | Optional | Without CAM |
| 41 | Optional | With CAM |

## 6-22.8 Configuration Assembly Data Attribute Format

The Configuration Assembly Data Attribute has the format shown below.

**Table 6-22.8 Configuration Assembly Data Format**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 40 | 0 | Value Bit Resolution | | | | | | | |
| | 1 | | | | | | | | |
| | 2 | Zero Offset | | | | | | | |
| | 3 | | | | | | | | |
| | 4 | | | | | | | | |
| 41 | 0 | Value Bit Resolution | | | | | | | |
| | 1 | | | | | | | | |
| | 2 | Zero Offset | | | | | | | |
| | 3 | | | | | | | | |
| | 4 | | | | | | | | |
| | 5 | | | | | | | | |
| | 6 | CAM Low Limit | | | | | | | |
| | 7 | | | | | | | | |
| | 8 | | | | | | | | |
| | 9 | | | | | | | | |
| | 10 | CAM High Limit | | | | | | | |
| | 11 | | | | | | | | |
| | 12 | | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-22.9 Mapping Configuration Assembly Data Attribute Components

The following table indicates the configuration Assembly Data Attribute mapping for the Resolver device.

**Table 6-22.9 Configuration Assembly Data Mapping**

| Data Component | Class | | Instance Number | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | | Name | Number |
| Resolution | Position Sensor | 23hex | 1 | Bit Resolution | 5 |
| Zero Offset | Position Sensor | 23hex | 1 | Zero Offset | 6 |
| CAM Low Limit | Position Sensor | 23hex | 1 | CAM Low | 7 |
| CAM High Limit | Position Sensor | 23hex | 1 | CAM High | 8 |

## 6-22.10 Defining Device Configuration

Public access to the Position Sensor Object by the Message Router must be supported for configuration of this device type. If supported, the optional Parameter Object may be used to access the device type's configuration parameters.

If the Parameter Object is supported it must support a minimum of the Parameter Stub attributes, and may optionally support any or all of the Full Parameter Object attributes.

### 6-22.10.1 Parameter Object Instances

The following table indicates the Parameter Object Instances supported by the Resolver device.

**Table 6-22.10 Parameter Object Instances Supported**

| Number | Name |
|---|---|
| 1 | Value Bit Resolution |
| 2 | Zero Offset |
| 3 | CAM Low Limit |
| 4 | CAM High Limit |

### 6-22.10.2 Mapping Parameter Object Data

The following table indicates the Parameter Object data mapping for the Resolver device.

**Table 6-22.11 Parameter Object Data Mapping**

| Configuration Parameter | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Resolution | Position Sensor | 23hex | 1 | Bit Resolution | 5 |
| Zero Offset | Position Sensor | 23hex | 1 | Zero Offset | 6 |
| CAM Low Limit | Position Sensor | 23hex | 1 | CAM Low Limit | 7 |
| CAM High Limit | Position Sensor | 23hex | 1 | CAM High Limit | 8 |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-22.10.3 Configuration Parameter Definitions

The following sections of an example EDS show the information necessary to define the configuration parameters for a Resolver device.

```
[ParamClass]
      MaxInst=4          $Maximum Instance Number
      Descriptor=0x09    $Parameter Class Descriptor
      CfgAssembly=2      $Configuration Assembly Instance

[Params]
      Param1=                       $Resolution parameter
        0,                          $Data placeholder
        6, "20 23 24 01 30 05",     $Path size and path to attribute
        0x0000,                     $Descriptor
        8, 1,                       $Data type and size (USINT)
        "Bit Resolution",           $Name
        "Bits",                     $Units
        "",                         $(not used)
        1, 32,  (vendor specific),  $Min, max and default values
        0, 0, 0, 0, 0, 0, 0, 0, 0;  $(not used)

      Param2=                       $Zero Offset parameter
        0,                          $Data placeholder
        6, "20 23 24 01 30 06",     $Path size and path to attribute
        0x0000,                     $Descriptor
        9, 4,                       $Data type and size (UDINT)
        "Zero Offset",              $Name
        "",                         $Units (none)
        "",                         $(not used)
        0, 0xFFFFFFFF, 0,           $Min, max and default values
        0, 0, 0, 0, 0, 0, 0, 0, 0;  $(not used)
      Param3=                       $CAM Low Limit
        0,                          $Data placeholder
        6, "20 23 24 01 30 07",     $Path size and path to attribute
        0x0000,                     $Descriptor
        9, 4,                       $Data type and size (UDINT)
        "CAM Low Limit",            $Name
        "",                         $Units (none)
        "",                         $(not used)
        0, 0xFFFFFFFF, 0,           $Min, max and default values
        0, 0, 0, 0, 0, 0, 0, 0, 0;  $(not used)
      Param4=                       $CAM High Limit
        0,                          $Data placeholder
        6, "20 23 24 01 30 08",     $Path size and path to attribute
        0x0000,                     $Descriptor
        9, 4,                       $Data type and size (UDINT)
        "CAM High Limit",           $Name
        "",                         $Units (none)
        "",                         $(not used)
        0, 0xFFFFFFFF, 0,           $Min, max and default values
        0, 0, 0, 0, 0, 0, 0, 0, 0;  $(not used)
```



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-22.11 Effect of Configuration Parameters on Behavior

The configuration parameters affect the device's behavior as shown below.

**Table 6-22.12 Configuration Parameter Effect on Behavior**

| Parameter | Effect on behavior |
|---|---|
| Bit Resolution | Sets the number of significant bits in the Value Attribute of the Position Sensor Object |
| Zero Offset | Sets the zero point for the Value Attribute of the Position Sensor Object |
| CAM Low | Sets the low threshold for the CAM Attribute of the Position Sensor Object |
| CAM High | Sets the high threshold for the CAM Attribute of the Position Sensor Object |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-23 Control Station Device

The profile for this device type will be defined by the ODVA.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-24 Message Display Device

The profile for this device type will be defined by the ODVA.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-25 Circuit Breaker

The profile for this device type will be defined by the ODVA.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-26 Pneumatic Valve Device

## Device Type 1B hex

This Device Profile defines minimum requirements for a Pneumatic Valve Manifold with one or more solenoid points and ability to support optional discrete input points.

## 6-26.1 Object Model

The Object Model is illustrated in Figure 6-26.1 and the object classes are described in the following table:

**Table 6-26.1 Objects Present in a Pneumatic Valve Device**

| Object Classes | Class ID | Required / Optional | # of Instances |
|---|---|---|---|
| CIP Required Objects | | | See Section 6-2.1 |
| Connection or Connection Manager | | Required | 2 or more * |
| Assembly | 0x04 | Required | 1 or more * |
| Discrete Input ** | 0x08 | Optional | * |
| Discrete Output *** | 0x09 | Required | 1 or more * |
| Parameter | 0x0F | Optional | Vendor Specific |

* Depends on the level of I/O support provided by the product

** Discrete Input Class includes optional solenoid status points and/or optional discrete input points.

*** Discrete Output Class includes the solenoid valve points.

**Figure 6-26.1 Object Model for a Pneumatic Valve Manifold**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-26.2 How Objects Affect Behavior

The objects for this device affect its behavior in the following manner:

**Table 6-26.2 Object Effect on Behavior**

| Object Classes | Effect on Behavior |
|---|---|
| Identity | Supports the Reset Service |
| Message Router | No affect |
| DeviceNet | Configures port attributes |
| Assembly | Defines I/O data format |
| Connection | Contains the number of logical ports |
| Discrete Input | Defines the behavior of solenoid status and discrete input points |
| Discrete Output | Defines the behavior of solenoid output points |
| Parameter | Provides a public interface to device configuration data |

## 6-26.3 Defining Object Interfaces

**Table 6-26.3 Object Interfaces**

| Object Classes | Interface(s) |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | Message Router or I/O Connection |
| Discrete Input | Message Router or Assembly Object |
| Discrete Output | Message Router or Assembly Object |
| Parameter | Message Router or Assembly Object |

## 6-26.4 I/O Assembly Instances

The following table defines the **Solenoid Status (Input type)** assembly instances:

**Table 6-26.4 I/O Assembly Instances – Solenoid Status**

| Instance Number | Type | Name |
|---|---|---|
| 1 | Input | 1 Solenoid Status |
| 2 | Input | 2 Solenoid Status Points |
| 3 | Input | 4 Solenoid Status Points |
| 4 | Input | 8 Solenoid Status Points |
| 5 | Input | 16 Solenoid Status Points |
| 6 | Input | 24 Solenoid Status Points |
| 7 | Input | 32 Solenoid Status Points |
| 8 | Input | 48 Solenoid Status Points |
| 9 | Input | 64 Solenoid Status Points |
| 10 | Input | L Solenoid Status Points |

The following table defines the **Discrete Input (Input type)** assembly instances:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-26.5 I/O Assembly Instances – Discrete Input**

| Instance Number | Type | Name |
|---|---|---|
| 11 | Input | 1 Discrete Input Point |
| 12 | Input | 2 Discrete Input Points |
| 13 | Input | 4 Discrete Input Points |
| 14 | Input | 8 Discrete Input Points |
| 15 | Input | 16 Discrete Input Points |
| 16 | Input | 24 Discrete Input Points |
| 17 | Input | 32 Discrete Input Points |
| 18 | Input | 48 Discrete Input Points |
| 19 | Input | 64 Discrete Input Points |
| 20 | Input | M Discrete Input Points |

The following table defines the Input type instances where **Solenoid Status points** and **Discrete Input** points are both present.

**Table 6-26.6 I/O Assembly Instances – Solenoid Status & Discrete Input**

| Instance Number | Type | Name |
|---|---|---|
| 21 | Input | 1 Solenoid Status Points + 1 Discrete Input Points |
| 22 | Input | 2 Solenoid Status Points + 2 Discrete Input Points |
| 23 | Input | 4 Solenoid Status Points + 4 Discrete Input Points |
| 24 | Input | 8 Solenoid Status Points + 8 Discrete Input Points |
| 25 | Input | 16 Solenoid Status Points + 16 Discrete Input Points |
| 26 | Input | 24 Solenoid Status Points + 24 Discrete Input Points |
| 27 | Input | 32 Solenoid Status Points + 32 Discrete Input Points |
| 28 | Input | 48 Solenoid Status Points + 48 Discrete Input Points |
| 29 | Input | 64 Solenoid Status Points + 64 Discrete Input Points |
| 30 | --- | Reserved by CIP |

The following table defines the **Solenoid Output (Output type)** assembly instances:



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-26.7 I/O Assembly Instances – Solenoid Output**

| Instance Number | Type | Name |
|---|---|---|
| 31 | Output | Single Solenoid Output Point |
| 32 | Output | 2 Solenoid Output Points |
| 33 | Output | 4 Solenoid Output Points |
| 34 | Output | 8 Solenoid Output Points |
| 35 | Output | 16 Solenoid Output Points |
| 36 | Output | 24 Solenoid Output Points |
| 37 | Output | 32 Solenoid Output Points |
| 38 | Output | 48 Solenoid Output Points |
| 39 | Output | 64 Solenoid Output Points |
| 40 | Output | N Solenoid Output Points |
| 41 ~ 99 | -- | Reserved by CIP |
| 100 ~ 199 | -- | Vendor Specific |

## 6-26.5 I/O Assembly Data Attribute Format

### 6-26.5.1 The I/O Assembly data attribute for *Solenoids Status Data* is shown below:

**Table 6-26.8 I/O Assembly Data Attributes – Solenoid Status Data**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | Reserved | | | | | | | Solenoid Status1 |
| | | | | | | | | | |
| 2 | 0 | Reserved | | | | | | Solenoid Status2 | Solenoid Status1 |
| | | | | | | | | | |
| 3 | 0 | Reserved | | | | Solenoid Status4 | Solenoid Status3 | Solenoid Status2 | Solenoid Status1 |
| | | | | | | | | | |
| 4 | 0 | Solenoid Status8 | Solenoid Status7 | Solenoid Status6 | Solenoid Status5 | Solenoid Status4 | Solenoid Status3 | Solenoid Status2 | Solenoid Status1 |
| | | | | | | | | | |
| 5 | 0 | Solenoid Status8 | Solenoid Status7 | Solenoid Status6 | Solenoid Status5 | Solenoid Status4 | Solenoid Status3 | Solenoid Status2 | Solenoid Status1 |
| | 1 | Solenoid Status16 | Solenoid Status15 | Solenoid Status14 | Solenoid Status13 | Solenoid Status12 | Solenoid Status11 | Solenoid Status10 | Solenoid Status9 |
| | | | | | | | | | |
| 6 | 0 | Solenoid Status8 | Solenoid Status7 | Solenoid Status6 | Solenoid Status5 | Solenoid Status4 | Solenoid Status3 | Solenoid Status2 | Solenoid Status1 |
| | 1 | Solenoid Status16 | Solenoid Status15 | Solenoid Status14 | Solenoid Status13 | Solenoid Status12 | Solenoid Status11 | Solenoid Status10 | Solenoid Status9 |
| | 2 | Solenoid Status24 | Solenoid Status23 | Solenoid Status22 | Solenoid Status21 | Solenoid Status20 | Solenoid Status19 | Solenoid Status18 | Solenoid Status17 |
| | | | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Pneumatic Valve Device, Type: $1B_{Hex}$

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 0 | Solenoid Status8 | Solenoid Status7 | Solenoid Status6 | Solenoid Status5 | Solenoid Status4 | Solenoid Status3 | Solenoid Status2 | Solenoid Status1 |
| | 1 | Solenoid Status16 | Solenoid Status15 | Solenoid Status14 | Solenoid Status13 | Solenoid Status12 | Solenoid Status11 | Solenoid Status10 | Solenoid Status9 |
| | 2 | Solenoid Status24 | Solenoid Status23 | Solenoid Status22 | Solenoid Status21 | Solenoid Status20 | Solenoid Status19 | Solenoid Status18 | Solenoid Status17 |
| | 3 | Solenoid Status32 | Solenoid Status31 | Solenoid Status30 | Solenoid Status29 | Solenoid Status28 | Solenoid Status27 | Solenoid Status26 | Solenoid Status25 |
| | | | | | | | | | |
| 8 | 0 | Solenoid Status8 | Solenoid Status7 | Solenoid Status6 | Solenoid Status5 | Solenoid Status4 | Solenoid Status3 | Solenoid Status2 | Solenoid Status1 |
| | 1 | Solenoid Status16 | Solenoid Status15 | Solenoid Status14 | Solenoid Status13 | Solenoid Status12 | Solenoid Status11 | Solenoid Status10 | Solenoid Status9 |
| | 2 | Solenoid Status24 | Solenoid Status23 | Solenoid Status22 | Solenoid Status21 | Solenoid Status20 | Solenoid Status19 | Solenoid Status18 | Solenoid Status17 |
| | 3 | Solenoid Status32 | Solenoid Status31 | Solenoid Status30 | Solenoid Status29 | Solenoid Status28 | Solenoid Status27 | Solenoid Status26 | Solenoid Status25 |
| | 4 | Solenoid Status40 | Solenoid Status39 | Solenoid Status38 | Solenoid Status37 | Solenoid Status36 | Solenoid Status35 | Solenoid Status34 | Solenoid Status33 |
| | 5 | Solenoid Status48 | Solenoid Status47 | Solenoid Status46 | Solenoid Status45 | Solenoid Status44 | Solenoid Status43 | Solenoid Status42 | Solenoid Status41 |
| | | | | | | | | | |
| 9 | 0 | Solenoid Status8 | Solenoid Status7 | Solenoid Status6 | Solenoid Status5 | Solenoid Status4 | Solenoid Status3 | Solenoid Status2 | Solenoid Status1 |
| | 1 | Solenoid Status16 | Solenoid Status15 | Solenoid Status14 | Solenoid Status13 | Solenoid Status12 | Solenoid Status11 | Solenoid Status10 | Solenoid Status9 |
| | 2 | Solenoid Status24 | Solenoid Status23 | Solenoid Status22 | Solenoid Status21 | Solenoid Status20 | Solenoid Status19 | Solenoid Status18 | Solenoid Status17 |
| | 3 | Solenoid Status32 | Solenoid Status31 | Solenoid Status30 | Solenoid Status29 | Solenoid Status28 | Solenoid Status27 | Solenoid Status26 | Solenoid Status25 |
| | 4 | Solenoid Status40 | Solenoid Status39 | Solenoid Status38 | Solenoid Status37 | Solenoid Status36 | Solenoid Status35 | Solenoid Status34 | Solenoid Status33 |
| | 5 | Solenoid Status48 | Solenoid Status47 | Solenoid Status46 | Solenoid Status45 | Solenoid Status44 | Solenoid Status43 | Solenoid Status42 | Solenoid Status41 |
| | 6 | Solenoid Status56 | Solenoid Status55 | Solenoid Status54 | Solenoid Status53 | Solenoid Status52 | Solenoid Status51 | Solenoid Status50 | Solenoid Status49 |
| | 7 | Solenoid Status64 | Solenoid Status63 | Solenoid Status62 | Solenoid Status61 | Solenoid Status60 | Solenoid Status59 | Solenoid Status58 | Solenoid Status57 |
| | | | | | | | | | |
| 10 | 0 | Solenoid Status8 | Solenoid Status7 | Solenoid Status6 | Solenoid Status5 | Solenoid Status4 | Solenoid Status3 | Solenoid Status2 | Solenoid Status1 |
| | • • • | | | | | | | | |
| | K | | | | | | Solenoid Status L | Solenoid Status L-1 | Solenoid Status L-2 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-26.5.2 The I/O Assembly data attribute for *General Purpose Discrete Input Data* is shown below:

**Table 6-26.9 I/O Assembly Data Attributes – Gen Purpose Discrete Input Data**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 11 | 0 | Reserved | | | | | | | Discrete Input 1 |
| | | | | | | | | | |
| 12 | 0 | Reserved | | | | | | Discrete Input2 | Discrete Input1 |
| | | | | | | | | | |
| 13 | 0 | Reserved | | | | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | | | | | | | | | |
| 14 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | | | | | | | | | |
| 15 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input6 | Discrete Input 15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input 9 |
| | | | | | | | | | |
| 16 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input16 | Discrete Input15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input 9 |
| | 2 | Discrete Input24 | Discrete Input23 | Discrete Input22 | Discrete Input21 | Discrete Input20 | Discrete Input19 | Discrete Input18 | Discrete Input17 |
| | | | | | | | | | |
| 17 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input16 | Discrete Input 15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input 9 |
| | 2 | Discrete Input24 | Discrete Input23 | Discrete Input22 | Discrete Input21 | Discrete Input20 | Discrete Input19 | Discrete Input18 | Discrete Input17 |
| | 3 | Discrete Input32 | Discrete Input31 | Discrete Input 30 | Discrete Input29 | Discrete Input28 | Discrete Input27 | Discrete Input26 | Discrete Input25 |
| | | | | | | | | | |
| 18 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input16 | Discrete Input 15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input 9 |
| | 2 | Discrete Input24 | Discrete Input23 | Discrete Input22 | Discrete Input21 | Discrete Input20 | Discrete Input19 | Discrete Input18 | Discrete Input17 |
| | 3 | Discrete Input32 | Discrete Input31 | Discrete Input 30 | Discrete Input29 | Discrete Input28 | Discrete Input27 | Discrete Input26 | Discrete Input25 |
| | 4 | Discrete Input40 | Discrete Input39 | Discrete Input38 | Discrete Input37 | Discrete Input 36 | Discrete Input35 | Discrete Input34 | Discrete Input33 |
| | 5 | Discrete Input48 | Discrete Input47 | Discrete Input46 | Discrete Input45 | Discrete Input44 | Discrete Input43 | Discrete Input42 | Discrete Input41 |
| | | | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Pneumatic Valve Device, Type: $1B_{Hex}$**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 19 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 1 | Discrete Input16 | Discrete Input 15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input 9 |
| | 2 | Discrete Input24 | Discrete Input23 | Discrete Input22 | Discrete Input21 | Discrete Input20 | Discrete Input19 | Discrete Input18 | Discrete Input17 |
| | 3 | Discrete Input32 | Discrete Input31 | Discrete Input30 | Discrete Input29 | Discrete Input28 | Discrete Input27 | Discrete Input26 | Discrete Input25 |
| | 4 | Discrete Input40 | Discrete Input39 | Discrete Input38 | Discrete Input37 | Discrete Input36 | Discrete Input35 | Discrete Input34 | Discrete Input33 |
| | 5 | Discrete Input48 | Discrete Input 47 | Discrete Input46 | Discrete Input45 | Discrete Input 44 | Discrete Input43 | Discrete Input42 | Discrete Input41 |
| | 6 | Discrete Input56 | Discrete Input55 | Discrete Input54 | Discrete Input53 | Discrete Input 52 | Discrete Input51 | Discrete Input50 | Discrete Input 49 |
| | 7 | Discrete Input 64 | Discrete Input63 | Discrete Input62 | Discrete Input61 | Discrete Input60 | Discrete Input 59 | Discrete Input58 | Discrete Input57 |
| | | | | | | | | | |
| 20 | 0 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | • • • | | | | | | | | |
| | K | | | | | | Discrete Input M | Discrete Input M-1 | Discrete Input M-2 |

### 6-26.5.3 The I/O Assembly data attribute for *Solenoids Status Data* plus *Discrete Input Data*

**Table 6-26.10 I/O Assembly Data Attributes – Sol. Status & Discrete Input Data**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 21 | 0 | Reserved | | | | | | Discrete Input1 | Solenoid Status1 |
| | | | | | | | | | |
| 22 | 0 | Reserved | | | | Discrete Input2 | Discrete Input1 | Solenoid Status2 | Solenoid Status1 |
| | | | | | | | | | |
| 23 | 0 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 | Solenoid Status4 | Solenoid Status3 | Solenoid Status2 | Solenoid Status1 |
| | | | | | | | | | |
| 24 | 0 | Solenoid Status8 | Solenoid Status7 | Solenoid Status6 | Solenoid Status5 | Solenoid Status4 | Solenoid Status3 | Solenoid Status2 | Solenoid Status1 |
| | 1 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | | | | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Pneumatic Valve Device, Type: $1B_{Hex}$

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 25 | 0 | Solenoid Status8 | Solenoid Status7 | Solenoid Status6 | Solenoid Status5 | Solenoid Status4 | Solenoid Status3 | Solenoid Status2 | Solenoid Status1 |
| | 1 | Solenoid Status16 | Solenoid Status15 | Solenoid Status14 | Solenoid Status13 | Solenoid Status12 | Solenoid Status11 | Solenoid Status10 | Solenoid Status9 |
| | 2 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 3 | Discrete Input16 | Discrete Input 15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input 9 |
| | | | | | | | | | |
| 26 | 0 | Solenoid Status8 | Solenoid Status7 | Solenoid Status6 | Solenoid Status5 | Solenoid Status4 | Solenoid Status3 | Solenoid Status2 | Solenoid Status1 |
| | 1 | Solenoid Status16 | Solenoid Status15 | Solenoid Status14 | Solenoid Status13 | Solenoid Status12 | Solenoid Status11 | Solenoid Status10 | Solenoid Status9 |
| | 2 | Solenoid Status24 | Solenoid Status23 | Solenoid Status22 | Solenoid Status21 | Solenoid Status20 | Solenoid Status19 | Solenoid Status18 | Solenoid Status17 |
| | 3 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 4 | Discrete Input16 | Discrete Input15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input9 |
| | 5 | Discrete Input24 | Discrete Input23 | Discrete Input22 | Discrete Input21 | Discrete Input20 | Discrete Input19 | Discrete Input18 | Discrete Input17 |
| | | | | | | | | | |
| 27 | 0 | Solenoid Status8 | Solenoid Status7 | Solenoid Status6 | Solenoid Status5 | Solenoid Status4 | Solenoid Status3 | Solenoid Status2 | Solenoid Status1 |
| | 1 | Solenoid Status16 | Solenoid Status15 | Solenoid Status14 | Solenoid Status13 | Solenoid Status12 | Solenoid Status11 | Solenoid Status10 | Solenoid Status9 |
| | 2 | Solenoid Status24 | Solenoid Status23 | Solenoid Status22 | Solenoid Status21 | Solenoid Status20 | Solenoid Status19 | Solenoid Status18 | Solenoid Status17 |
| | 3 | Solenoid Status32 | Solenoid Status31 | Solenoid Status30 | Solenoid Status29 | Solenoid Status28 | Solenoid Status27 | Solenoid Status26 | Solenoid Status25 |
| | 4 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 5 | Discrete Input16 | Discrete Input 15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input 9 |
| | 6 | Discrete Input24 | Discrete Input23 | Discrete Input22 | Discrete Input21 | Discrete Input20 | Discrete Input19 | Discrete Input18 | Discrete Input17 |
| | 7 | Discrete Input32 | Discrete Input31 | Discrete Input 30 | Discrete Input29 | Discrete Input28 | Discrete Input27 | Discrete Input26 | Discrete Input25 |
| | | | | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Pneumatic Valve Device, Type: $1B_{Hex}$

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 28 | 0 | Solenoid Status8 | Solenoid Status7 | Solenoid Status6 | Solenoid Status5 | Solenoid Status4 | Solenoid Status3 | Solenoid Status2 | Solenoid Status1 |
| | 1 | Solenoid Status16 | Solenoid Status15 | Solenoid Status14 | Solenoid Status13 | Solenoid Status12 | Solenoid Status11 | Solenoid Status10 | Solenoid Status9 |
| | 2 | Solenoid Status24 | Solenoid Status23 | Solenoid Status22 | Solenoid Status21 | Solenoid Status20 | Solenoid Status19 | Solenoid Status18 | Solenoid Status17 |
| | 3 | Solenoid Status32 | Solenoid Status31 | Solenoid Status30 | Solenoid Status29 | Solenoid Status28 | Solenoid Status27 | Solenoid Status26 | Solenoid Status25 |
| | 4 | Solenoid Status40 | Solenoid Status39 | Solenoid Status38 | Solenoid Status37 | Solenoid Status36 | Solenoid Status35 | Solenoid Status34 | Solenoid Status33 |
| | 5 | Solenoid Status48 | Solenoid Status47 | Solenoid Status46 | Solenoid Status45 | Solenoid Status44 | Solenoid Status43 | Solenoid Status42 | Solenoid Status41 |
| | 6 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input1 |
| | 7 | Discrete Input16 | Discrete Input 15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input 9 |
| | 8 | Discrete Input24 | Discrete Input23 | Discrete Input22 | Discrete Input21 | Discrete Input20 | Discrete Input19 | Discrete Input18 | Discrete Input17 |
| | 9 | Discrete Input32 | Discrete Input31 | Discrete Input30 | Discrete Input29 | Discrete Input28 | Discrete Input27 | Discrete Input26 | Discrete Input25 |
| | 10 | Discrete Input40 | Discrete Input39 | Discrete Input38 | Discrete Input37 | Discrete Input36 | Discrete Input35 | Discrete Input34 | Discrete Input33 |
| | 11 | Discrete Input48 | Discrete Input 47 | Discrete Input46 | Discrete Input45 | Discrete Input 44 | Discrete Input43 | Discrete Input42 | Discrete Input41 |
| | | | | | | | | | |
| *29* | 0 | Solenoid Status8 | Solenoid Status7 | Solenoid Status6 | Solenoid Status5 | Solenoid Status4 | Solenoid Status3 | Solenoid Status2 | Solenoid Status1 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Pneumatic Valve Device, Type: $1B_{Hex}$**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | Solenoid Status16 | Solenoid Status15 | Solenoid Status14 | Solenoid Status13 | Solenoid Status12 | Solenoid Status11 | Solenoid Status10 | Solenoid Status9 |
| | 2 | Solenoid Status24 | Solenoid Status23 | Solenoid Status22 | Solenoid Status21 | Solenoid Status20 | Solenoid Status19 | Solenoid Status18 | Solenoid Status17 |
| | 3 | Solenoid Status32 | Solenoid Status31 | Solenoid Status30 | Solenoid Status29 | Solenoid Status28 | Solenoid Status27 | Solenoid Status26 | Solenoid Status25 |
| | 4 | Solenoid Status40 | Solenoid Status39 | Solenoid Status38 | Solenoid Status37 | Solenoid Status36 | Solenoid Status35 | Solenoid Status34 | Solenoid Status33 |
| | 5 | Solenoid Status48 | Solenoid Status47 | Solenoid Status46 | Solenoid Status45 | Solenoid Status44 | Solenoid Status43 | Solenoid Status42 | Solenoid Status41 |
| | 6 | Solenoid Status56 | Solenoid Status55 | Solenoid Status54 | Solenoid Status53 | Solenoid Status52 | Solenoid Status51 | Solenoid Status50 | Solenoid Status49 |
| | 7 | Solenoid Status64 | Solenoid Status63 | Solenoid Status62 | Solenoid Status61 | Solenoid Status60 | Solenoid Status59 | Solenoid Status58 | Solenoid Status57 |
| | 8 | Discrete Input8 | Discrete Input7 | Discrete Input6 | Discrete Input5 | Discrete Input4 | Discrete Input3 | Discrete Input2 | Discrete Input 1 |
| | 9 | Discrete Input16 | Discrete Input15 | Discrete Input14 | Discrete Input13 | Discrete Input12 | Discrete Input11 | Discrete Input10 | Discrete Input9 |
| | 10 | Discrete Input24 | Discrete Input23 | Discrete Input22 | Discrete Input 21 | Discrete Input 20 | Discrete Input19 | Discrete Input18 | Discrete Input17 |
| | 11 | Discrete Input32 | Discrete Input31 | Discrete Input30 | Discrete Input29 | Discrete Input28 | Discrete Input27 | Discrete Input26 | Discrete Input25 |
| | 12 | Discrete Input40 | Discrete Input39 | Discrete Input38 | Discrete Input37 | Discrete Input36 | Discrete Input35 | Discrete Input34 | Discrete Input33 |
| | 13 | Discrete Input48 | Discrete Input 47 | Discrete Input46 | Discrete Input45 | Discrete Input 44 | Discrete Input43 | Discrete Input42 | Discrete Input41 |
| | 14 | Discrete Input56 | Discrete Input55 | Discrete Input54 | Discrete Input53 | Discrete Input 52 | Discrete Input51 | Discrete Input50 | Discrete Input 49 |
| | 15 | Discrete Input 64 | Discrete Input63 | Discrete Input62 | Discrete Input61 | Discrete Input60 | Discrete Input 59 | Discrete Input58 | Discrete Input57 |

### 6-26.5.4 The I/O Assembly data attribute for the *Solenoid Valves Output Data*

**Table 6-26.11 I/O Assembly Data Attributes – Solenoid Valves Output Data**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 31 | 0 | Reserved | | | | | | | Solenoid Output1 |
| | | | | | | | | | |
| 32 | 0 | Reserved | | | | | | Solenoid Output2 | Solenoid Output1 |
| | | | | | | | | | |
| 33 | 0 | Reserved | | | | Solenoid Output4 | Solenoid Output3 | Solenoid Output2 | Solenoid Output1 |
| | | | | | | | | | |
| 34 | 0 | Solenoid Output8 | Solenoid Output7 | Solenoid Output6 | Solenoid Output5 | Solenoid Output4 | Solenoid Output3 | Solenoid Output2 | Solenoid Output1 |
| | | | | | | | | | |
| 35 | 0 | Solenoid Output8 | Solenoid Output7 | Solenoid Output6 | Solenoid Output5 | Solenoid Output4 | Solenoid Output3 | Solenoid Output2 | Solenoid Output1 |
| | 1 | Solenoid Output16 | Solenoid Output15 | Solenoid Output14 | Solenoid Output13 | Solenoid Output12 | Solenoid Output11 | Solenoid Output10 | Solenoid Output9 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Pneumatic Valve Device, Type: $1B_{Hex}$

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| 36 | 0 | Solenoid Output8 | Solenoid Output7 | Solenoid Output6 | Solenoid Output5 | Solenoid Output4 | Solenoid Output3 | Solenoid Output2 | Solenoid Output1 |
| | 1 | Solenoid Output16 | Solenoid Output15 | Solenoid Output14 | Solenoid Output13 | Solenoid Output12 | Solenoid Output11 | Solenoid Output10 | Solenoid Output9 |
| | 2 | Solenoid Output24 | Solenoid Output23 | Solenoid Output22 | Solenoid Output21 | Solenoid Output20 | Solenoid Output19 | Solenoid Output18 | Solenoid Output17 |
| | | | | | | | | | |
| 37 | 0 | Solenoid Output8 | Solenoid Output7 | Solenoid Output6 | Solenoid Output5 | Solenoid Output4 | Solenoid Output3 | Solenoid Output2 | Solenoid Output1 |
| | 1 | Solenoid Output16 | Solenoid Output15 | Solenoid Output14 | Solenoid Output13 | Solenoid Output12 | Solenoid Output11 | Solenoid Output10 | Solenoid Output9 |
| | 2 | Solenoid Output24 | Solenoid Output23 | Solenoid Output22 | Solenoid Output21 | Solenoid Output20 | Solenoid Output19 | Solenoid Output18 | Solenoid Output17 |
| | 3 | Solenoid Output32 | Solenoid Output31 | Solenoid Output30 | Solenoid Output29 | Solenoid Output28 | Solenoid Output27 | Solenoid Output26 | Solenoid Output25 |
| | | | | | | | | | |
| 38 | 0 | Solenoid Output8 | Solenoid Output7 | Solenoid Output6 | Solenoid Output5 | Solenoid Output4 | Solenoid Output3 | Solenoid Output2 | Solenoid Output1 |
| | 1 | Solenoid Output16 | Solenoid Output15 | Solenoid Output14 | Solenoid Output13 | Solenoid Output12 | Solenoid Output11 | Solenoid Output10 | Solenoid Output9 |
| | 2 | Solenoid Output24 | Solenoid Output23 | Solenoid Output22 | Solenoid Output21 | Solenoid Output20 | Solenoid Output19 | Solenoid Output18 | Solenoid Output17 |
| | 3 | Solenoid Output32 | Solenoid Output31 | Solenoid Output30 | Solenoid Output29 | Solenoid Output28 | Solenoid Output27 | Solenoid Output26 | Solenoid Output25 |
| | 4 | Solenoid Output40 | Solenoid Output39 | Solenoid Output38 | Solenoid Output37 | Solenoid Output36 | Solenoid Output35 | Solenoid Output34 | Solenoid Output33 |
| | 5 | Solenoid Output48 | Solenoid Output47 | Solenoid Output46 | Solenoid Output45 | Solenoid Output44 | Solenoid Output43 | Solenoid Output42 | Solenoid Output41 |
| | | | | | | | | | |
| 39 | 0 | Solenoid Output8 | Solenoid Output7 | Solenoid Output6 | Solenoid Output5 | Solenoid Output4 | Solenoid Output3 | Solenoid Output2 | Solenoid Output1 |
| | 1 | Solenoid Output16 | Solenoid Output15 | Solenoid Output14 | Solenoid Output13 | Solenoid Output12 | Solenoid Output11 | Solenoid Output10 | Solenoid Output9 |
| | 2 | Solenoid Output24 | Solenoid Output23 | Solenoid Output22 | Solenoid Output21 | Solenoid Output20 | Solenoid Output19 | Solenoid Output18 | Solenoid Output17 |
| | 3 | Solenoid Output32 | Solenoid Output31 | Solenoid Output30 | Solenoid Output29 | Solenoid Output28 | Solenoid Output27 | Solenoid Output26 | Solenoid Output25 |
| | 4 | Solenoid Output40 | Solenoid Output39 | Solenoid Output38 | Solenoid Output37 | Solenoid Output36 | Solenoid Output35 | Solenoid Output34 | Solenoid Output33 |
| | 5 | Solenoid Output48 | Solenoid Output47 | Solenoid Output46 | Solenoid Output45 | Solenoid Output44 | Solenoid Output43 | Solenoid Output42 | Solenoid Output41 |
| | 6 | Solenoid Output56 | Solenoid Output55 | Solenoid Output54 | Solenoid Output53 | Solenoid Output52 | Solenoid Output51 | Solenoid Output50 | Solenoid Output49 |
| | 7 | Solenoid Output64 | Solenoid Output63 | Solenoid Output62 | Solenoid Output61 | Solenoid Output60 | Solenoid Output59 | Solenoid Output58 | Solenoid Output57 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Pneumatic Valve Device, Type: $1B_{Hex}$**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 40 | 0 | Solenoid Output8 | Solenoid Output7 | Solenoid Output6 | Solenoid Output5 | Solenoid Output4 | Solenoid Output3 | Solenoid Output2 | Solenoid Output1 |
| | • • • | | | | | | | | |
| | K | | | | | | Solenoid Output N | Solenoid Output N-1 | Solenoid Output N-2 |

## 6-26.6 Mapping I/O Assembly Data Attribute Components

The following table indicates the I/O assembly Data attribute mapping for the Pneumatic Valve Manifold device for the input assemblies numbered 1 through 10. These assemblies only present Solenoid Status information. The exact meaning of a “Solenoid Status” point is defined and assigned in the vendor instruction manual. Unless otherwise specified, the value 1=TRUE=”Energized Solenoid” is assumed in each assembly. Likewise, the value 0=FALSE=”De-energized Solenoid” is assumed in each assembly unless otherwise specified.

**Table 6-26.12 I/O Assembly Data Mapping – Input Assemblies 1-10**

| Data Component | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Solenoid Status L | Discrete Input Point | 08hex | L | Value | 3 |

The following table indicates the I/O assembly Data attribute mapping for Pneumatic Valve Manifold device for the input assemblies numbered 11 through 20. These assemblies only present values from General Purpose Discrete Input points. These points are not explicitly associated with Solenoid Value status (as defined in assemblies 1 through 10). Rather, the exact meaning of each point shall be defined and assigned by the vendor instruction manual.

**Table 6-26.13 I/O Assembly Data Mapping – Input Assemblies 11-20**

| Data Component | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Discrete Input M | Discrete Input Point | 08hex | M | Value | 3 |

The following table indicates the I/O assembly Data attribute mapping for the Pneumatic Valve Manifold device for the input assemblies numbered 21 through 29. These assemblies present both Solenoid Status information as well as General Purpose Discrete Input point values. The exact meaning of a “Solenoid Status” point is defined and assigned in the vendor instruction manual. Unless otherwise specified, the value 1=TRUE=”Energized Solenoid” is assumed in each assembly. Likewise, the value 0=FALSE=”De-energized Solenoid” is assumed in each assembly unless otherwise specified. The Discrete Input point values are not explicitly associated with Solenoid Value status. The exact meaning of each point shall be defined and assigned by the vendor instruction manual.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-26.14 I/O Assembly Data Mapping – Input Assemblies 21-29**

| Data Component | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Solenoid Status L | Discrete Input Point | 08hex | L | Value | 3 |
| Discrete Input M | Discrete Input Point | 08hex | M | Value | 3 |

The following table indicates the I/O assembly Data attribute mapping for Pneumatic Valve Manifold device for the Solenoid output assemblies.

**Table 6-26.15 I/O Assembly Data Mapping – Solenoid Output Assemblies**

| Data Component | Class | | Instance | Attribute | |
|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number |
| Solenoid Output L | Discrete Input Point | 08hex | N | Value | 3 |

**Important:** If I/O Assembly instances 10 or 20 are supported, the “Max Instance” attribute at the class level of the Discrete Input Point class must be supported. If I/O Assembly instance 40 is supported, the “Max Instance” attribute at the class level of the Discrete Output Point class must be supported.

## 6-26.7 Configuration Data and Public Interfaces

Device configuration data and its public interfaces are vendor specific.

## 6-26.8 Parameter Object Instances

The Parameter Object Class, its instances and data mapping are optional and vendor specific.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-27 Contactor Device

**Device Type: 15 hex**

The Contactor device profile is part of a "Hierarchy of Motor Control Devices" that are supported by CIP. This hierarchy includes:

- Contactors, Overloads, and Across the Line Motor Starters
- Softstarters
- AC/DC Drives
- Servo Drives

Devices within this hierarchy use a common Control Supervisor object to control state behavior of the device. Devices within this hierarchy also support a hierarchy of "IO Assembly Instance" definitions which are used to pass control and status information to and from a device. Assembly instances are numbered so that each device type is assigned a range of instance numbers, with higher functionality devices supporting higher instance numbers. Devices within the hierarchy can choose to support some instance numbers that are lower than theirs in the hierarchy. For example, an AC Drive may choose to support some instances that are defined for Across the Line Motor Starters. This makes it easier to interchange drives and starters within a system.

This profile makes Motor Starters of the same device type inter-operable, but not directly interchangeable without doing configuration through a unit's local interface, a network configuration tool or other means of configuring outside the CIP interface.

## 6-27.1 Object Model

The Object Model in Figure 6-27.1 represents a Contactor. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-27.1 Objects Present in a Contactor Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| Message Router | Optional | - |
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Optional | 1 |
| Parameter | Optional | - |
| Control Supervisor | Required | 1 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-27.1 Object Model for a Contactor Device**

## 6-27.2 How Objects Affect Behavior

The objects for this device affect the device's behavior as shown in the table below.

**Table 6-27.2 Object Effect On Behavior**

| Object | Effect on behavior |
|---|---|
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines I/O data format |
| Parameter | Provides a public interface to device configuration data |
| Control Supervisor | Manages motor functions and operational states |

## 6-27.3 Defining Object Interfaces

The objects in the Contactor Device have the interfaces listed in the following table:

**Table 6-27.3 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| Parameter | Message Router |
| Control Supervisor | Message Router, Assembly or Parameter Object |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-27.4 Contactor Interface and Behavior

**Figure 6-27.2 Object Interfaces and Effect on Behavior**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-27.5 I/O Assembly Instance

The IO Assembly Instance definitions in this section define the format of the "data" attribute (attribute 3) for IO Assembly Instances. Through the use of predefined instance definitions, IO Assemblies support a hierarchy of motor control devices. The device hierarchy includes motor starters, soft starters, AC and DC drives, and servo drives. Assembly Instances are numbered within the hierarchy so that each device type is assigned a range of Assembly Instance numbers, with higher functionality devices supporting higher instance numbers. **Devices in the hierarchy can choose to support instance numbers that are lower than theirs in the hierarchy.** For example a Softstart may choose to support some IO Assemblies that are defined for Overload. The following table shows the Assembly Instance numbering for the motor control device hierarchy.

**Table 6-27.4 I/O Assembly Instances – Motor Control Device**

| Profile | I/O Type | Instance Range |
|---|---|---|
| Contactors, Overloads and | Output | 1-19 |
| Starters | Input | 50-69 |
| AC/DC Drive | Output | 20-29 |
| | Input | 70-79 |
| Servo Drive | Output | 30-49 |
| | Input | 80-99 |

The following IO Assembly Instances are defined for Contactors.

**Table 6-27.5 I/O Assembly Instances – Contactor Device**

| Instance | Type | Name |
|---|---|---|
| 1 | Output | Basic Contactor |
| 4 | Output | Extended Contactor |

If a bit is not used in an IO Assembly, it is reserved for use in other Assemblies. Reserved bits in Output Assemblies are ignored by the consuming device. Reserved bits in Input Assemblies are set to zero by the producing device.

### 6-27.5.1 Connection Paths to I/O Assembly Instances

The IO Assembly Instances are chosen for IO Connections by setting the "produced_connection_path" (attribute 14) and "consumed_connection_path" (attribute 16) attributes in the appropriate connection object.

Motor Control Devices use the Symbolic Segment Type (see Appendix C) to specify paths to the IO Assembly Instances in the Motor Control Hierarchy. IO Assembly Instances are represented by ASCII strings that contain the hex number of the Assembly Instance whose path is to be chosen.

The following example shows the Symbolic Segment used to specify Output Assembly Instance 20 (14 hex).



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-27.3 Symbolic Segment Example**

| Segment Type | | | | | | | | | First Character | | | | | | | | | Second Character | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| Segment type (symbolic) | | | Symbol size in bytes (2 bytes) | | | | | | ASCII 1 (31 hex) | | | | | | | | | ASCII 4 (34 hex) | | | | | | | |

## 6-27.6 I/O Assembly Data Attribute Format

**Table 6-27.6 I/O Assembly Data Attribute Format – Instance 1, Basic Contactor**

| **Instance 1: Basic Contactor**<br>This is the only required output assembly for device types Motor Contactor (0x15hex) and Softstart (0x15hex). | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Run1 |

**Table 6-27.7 I/O Assembly Data Attribute Format – Instance 4, Ext. Contactor**

| **Instance 4: Extended Contactor (see table for functional assignments)**<br>This assembly uses some optional attributes.. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Run2 | Run1 |

## 6-27.7 Mapping I/O Assembly Data Attribute Components

The following table indicates the I/O Assembly Data Attribute mapping for Contactor Output Assemblies.

**Table 6-27.8 I/O Assembly Data Attribute Mapping**

| Data Name | Class Name | Class Number | Instance | Attribute Name | Attribute Number |
|---|---|---|---|---|---|
| Run1 | Control Supervisor | 0x 29 | 1 | Run1 | 3 |
| Run2 | Control Supervisor | 0x 29 | 1 | Run2 | 4 |

## 6-27.8 Defining Device Configuration

Public access to the Control Supervisor Object must be supported for configuration of Contactor devices. If supported, optional Parameter Objects may be used to access the various configuration attributes in the Control Supervisor Object.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-28 Motor Starter Device

**Device Type: 16 hex**

The Motor Starter device profile is part of a "Hierarchy of Motor Control Devices" that are supported by CIP. This hierarchy includes:

- Contactors, Overloads, and Across the Line Motor Starters
- Softstarters
- AC/DC Drives
- Servo Drives

Devices within this hierarchy use a common Control Supervisor object to control state behavior of the device. Devices within this hierarchy also support a hierarchy of "IO Assembly Instance" definitions which are used to pass control and status information to and from a device. Assembly instances are numbered so that each device type is assigned a range of instance numbers, with higher functionality devices supporting higher instance numbers. Devices within the hierarchy can choose to support some instance numbers that are lower than theirs in the hierarchy. For example, an AC Drive may choose to support some instances that are defined for Across the Line Motor Starters. This makes it easier to interchange drives and starters within a system.

This profile makes Motor Starters of the same device type inter-operable, but not directly interchangeable without doing configuration through a unit's local interface, a network configuration tool or other means of configuring outside the CIP interface.

## 6-28.1 Object Model

The Object Model in Figure 6-28.1 represents a Motor Starter. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-28.1 Objects Present in a Motor Starter Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| Message Router | Optional | 1 |
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Optional | 1 |
| Parameter | Optional | - |
| Control Supervisor | Required | 1 |
| Overload | Required | - |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-28.1 Object Model for Motor Starter Device**

## 6-28.2 How Objects Affect Behavior

The objects for this device affect the device's behavior as shown in the table below.

**Table 6-28.2 Object Effect On Behavior**

| Object | Effect on behavior |
|---|---|
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines I/O data format |
| Parameter | Provides a public interface to device configuration data |
| Control Supervisor | Manages motor functions and operational states |
| Overload | Implements overload |

## 6-28.3 Defining Object Interfaces

The objects in the Motor Overload have the interfaces listed in the following table:

**Table 6-28.3 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| Parameter | Message Router |
| Control Supervisor | Message Router, Assembly or Parameter Object |
| Overload | Message Router, Assembly or Parameter Object |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-28.3.1 Starter Interface and Behavior

**Figure 6-28.2 Starter Interface & Behavior Diagram**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-28.3.2 Reversing Motor Starter Interface and Behavior

**Figure 6-28.3 Reversing Motor Starter Interface & Behavior Diagram**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-28.3.3 Two Speed Motor Starter Interface and Behavior

**Figure 6-28.4 Two Speed Motor Starter Interface & Behavior Diagram**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Motor Starter Device, Type: $16_{Hex}$

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-28.4 I/O Assembly Instances

The IO Assembly Instance definitions in this section define the format of the "data" attribute (attribute 3) for IO Assembly Instances. Through the use of predefined instance definitions, IO Assemblies support a hierarchy of motor control devices. The device hierarchy includes motor starters, soft starters, AC and DC drives, and servo drives. Assembly Instances are numbered within the hierarchy so that each device type is assigned a range of Assembly Instance numbers, with higher functionality devices supporting higher instance numbers. **Devices in the hierarchy can choose to support instance numbers that are lower than theirs in the hierarchy.** For example a Softstart may choose to support some IO Assemblies that are defined for Overload. The following table shows the Assembly Instance numbering for the motor control device hierarchy.

**Table 6-28.4 Assembly Instance Numbering for Motor Control Device Hierarchy**

| Profile | I/O Type | Instance Range |
|---|---|---|
| Contactors, Overloads and Starters | Output | 1-19 |
| | Input | 50-69 |
| AC/DC Drive | Output | 20-29 |
| | Input | 70-79 |
| Servo Drive | Output | 30-49 |
| | Input | 80-99 |

The following IO Assembly Instances are defined for Motor Starters.

**Table 6-28.5 Assembly Instances for Motor Starters**

| Instance | Type | Name |
|---|---|---|
| 3 | Output | Basic Motor Starter |
| 4 | Output | Extended Contactor |
| 5 | Output | Extended Motor Starter |
| | | |
| 52 | Input | Basic Motor Starter |
| 53 | Input | Extended Motor Starter 1 |
| 54 | Input | Extended Motor Starter 2 |

If a bit is not used in an IO Assembly, it is reserved for use in other Assemblies. Reserved bits in Output Assemblies are ignored by the consuming device. Reserved bits in Input Assemblies are set to zero by the producing device.

### 6-28.4.1 Connection Paths to I/O Assembly Instances

The IO Assembly Instances are chosen for IO Connections by setting the "produced_connection_path" (attribute 14) and "consumed_connection_path" (attribute 16) attributes in the appropriate connection object.

Motor Control Devices use the Symbolic Segment Type (see Appendix C) to specify paths to the IO Assembly Instances in the Motor Control Hierarchy. IO Assembly Instances are represented by ASCII strings that contain the hex number of the Assembly Instance whose path is to be chosen.

The following example shows the Symbolic Segment used to specify Output Assembly Instance 20 (14 hex).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-28.5 Symbolic Segment Example**

| Segment Type | | | | | | | | | First Character | | | | | | | | | Second Character | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| Segment type (symbolic) | | | Symbol size in bytes (2 bytes) | | | | | | ASCII 1 (31 hex) | | | | | | | | | ASCII 4 (34 hex) | | | | | | | |

## 6-28.5 I/O Assembly Data Attribute Format

### 6-28.5.1 Output Assembly Data Attribute Format

**Table 6-28.6 Instance 3: Basic Motor Starter**

| This is the only required output assembly for device type Motor Starter (16hex) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit .5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | Reserved | Reserved | Reserved | Reserved | Reserved | FaultReset | Reserved | Run1 |

**Table 6-28.7 Instance 4: Extended Contactor (see table for functional assignments)**

| This assembly uses some optional attributes. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Run2 | Run1 |

**Table 6-28.8 Instance 5: Extended Motor Starter (see table for functional assignments)**

| This assembly uses some optional attributes. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | Reserved | Reserved | Reserved | Reserved | Reserved | FaultReset | Run2 | Run1 |

### 6-28.5.2 Input Assembly Data Attribute Format

**Table 6-28.9 Instance 52: Basic Motor Starter**

| This is the only required input assembly for Motor Starter (16hex) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Running1 | Reserved | Faulted/ Trip |

**Table 6-28.10 Instance 53: Extended Motor Starter 1 (see table for functional assignments)**

| This assembly uses some optional attributes | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | Reserved | Reserved | Cntrlfrom Net | Ready | Reserved | Running1 | Warning | Faulted/ Trip |

**Table 6-28.11 Instance 54: Extended Motor Starter 2 (see table for functional assignments)**

| This assembly uses some optional attributes | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | Reserved | Reserved | Cntrlfrom Net | Ready | Running2 | Running1 | Warning | Faulted/ Trip |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-28.6 Mapping I/O Assembly Data Attribute Components

### 6-28.6.1 Mapping for Motor Starter Output Assembly Data Components

**Table 6-28.12 Output Data Assembly Mapping**

| Data Name | Class Name | Class Number | Instance | Attribute Name | Attribute Number |
|---|---|---|---|---|---|
| Run1 | Control Supervisor | 0x29 | 1 | Run1 | 3 |
| Run2 | Control Supervisor | 0x29 | 1 | Run2 | 4 |
| Fault Reset | Control Supervisor | 0x29 | 1 | FaultRst | 12 |

### 6-28.6.2 Mapping for Motor Starter Input Assembly Data Components

**Table 6-28.13 Input Data Assembly Mapping**

| Data Name | Class Name | Class Number | Instance | Attribute Name | Attribute Number |
|---|---|---|---|---|---|
| Faulted/ Trip | Control Supervisor | 0x29 | 1 | Faulted | 10 |
| Warning | Control Supervisor | 0x29 | 1 | Warning | 11 |
| Running1 | Control Supervisor | 0x29 | 1 | Running1 | 7 |
| Running2 | Control Supervisor | 0x29 | 1 | Running2 | 8 |
| Ready | Control Supervisor | 0x29 | 1 | Ready | 9 |
| Control From Net | Control Supervisor | 0x29 | 1 | CtrlFromNet | 15 |

## 6-28.7 Defining Device Configuration

Public access to the Control Supervisor Object and the Overload Object must be supported for configuration of Motor Starter devices. If supported, optional Parameter Objects may be used to access the various configuration attributes in the Control Supervisor Object and the Overload Object.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-29 Softstart Starter Device

## Device Type: 17 Hex

The Softstart Starter device profile is part of a "Hierarchy of Motor Control Devices" that are supported by CIP. This hierarchy includes:

- Contactors, Overloads, and Across the Line Motor Starters
- Softstarters
- AC/DC Drives
- Servo Drives

Devices within this hierarchy use a common Control Supervisor object to control state behavior of the device. Devices within this hierarchy also support a hierarchy of "IO Assembly Instance" definitions which are used to pass control and status information to and from a device. Assembly instances are numbered so that each device type is assigned a range of instance numbers, with higher functionality devices supporting higher instance numbers. Devices within the hierarchy can choose to support some instance numbers that are lower than theirs in the hierarchy. For example, an AC Drive may choose to support some instances that are defined for Across the Line Motor Starters. This makes it easier to interchange drives and starters within a system.

This profile makes Softstart Starters of the same device type inter-operable, but not directly interchangeable without doing configuration through a unit's local interface, a network configuration tool or other means of configuring outside the CIP interface.

## 6-29.1 Object Model

The Object Model in Figure 6-29.1 represents a Softstart Starter. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-29.1 Objects Present in a Softstart Starter Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| Message Router | Optional | 1 |
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Optional | 1 |
| Parameter | Optional | - |
| Control Supervisor | Required | 1 |
| Softstart | Optional | - |
| Overload | Optional | - |
| Motor Data | Optional | - |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-29.1 Object Model for Softstart Starter Device**

## 6-29.2 How Objects Affect Behavior

The objects for this device affect the device's behavior as shown in the table below.

**Table 6-29.2 Object Effect on Behavior**

| Object | Effect on behavior |
|---|---|
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines I/O data format |
| Parameter | Provides a public interface to device configuration data |
| Control Supervisor | Manages motor functions and operational states |
| Softstart | Implements the Softstart functions |
| Overload | Implements overload |
| Motor Data | Define motor data for motor connected to this device. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-29.3 Defining Object Interfaces

The objects in the Motor Overload have the interfaces listed in the following table:

**Table 6-29.3 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| Parameter | Message Router |
| Control Supervisor | Message Router, Assembly or Parameter Object |
| Softstart | Message Router or Assembly |
| Overload | Message Router, Assembly or Parameter Object |
| Motor Data | Message Router, Parameter Object |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-29.4 Softstart Motor Interface and Behavior

**Figure 6-29.2 Softstart Motor Interface & Behavior Diagram**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Softstart Starter Device, Type: $17_{Hex}$

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-29.5 I/O Assembly Instances

The IO Assembly Instance definitions in this section define the format of the "data" attribute (attribute 3) for IO Assembly Instances. Through the use of predefined instance definitions, IO Assemblies support a hierarchy of motor control devices. The device hierarchy includes motor starters, soft starters, AC and DC drives, and servo drives. Assembly Instances are numbered within the hierarchy so that each device type is assigned a range of Assembly Instance numbers, with higher functionality devices supporting higher instance numbers. **Devices in the hierarchy can choose to support instance numbers that are lower than theirs in the hierarchy.** For example a Softstart may choose to support some IO Assemblies that are defined for Overload. The following table shows the Assembly Instance numbering for the motor control device hierarchy.

**Table 6-29.4 I/O Assembly Instance Numbering Ranges**

| Profile | I/O Type | Instance Range |
|---|---|---|
| Contactors, Overloads and | Output | 1-19 |
| Starters | Input | 50-69 |
| AC/DC Drive | Output | 20-29 |
| | Input | 70-79 |
| Servo Drive | Output | 30-49 |
| | Input | 80-99 |

The following IO Assembly Instances are defined for Softstarters.

**Table 6-29.5 I/O Assembly Instances for Softstarters**

| Instance | Type | Name |
|---|---|---|
| 60 | Input | Basic Softstart |
| 61 | Input | Extended Softstart |

### 6-29.5.1 Connection Paths to I/O Assembly Instances

The IO Assembly Instances are chosen for IO Connections by setting the "produced_connection_path" (attribute 14) and "consumed_connection_path" (attribute 16) attributes in the appropriate connection object.

Motor Control Devices use the Symbolic Segment Type (see Appendix C) to specify paths to the IO Assembly Instances in the Motor Control Hierarchy. IO Assembly Instances are represented by ASCII strings that contain the hex number of the Assembly Instance whose path is to be chosen.

The following example shows the Symbolic Segment used to specify Output Assembly Instance 20 (14 hex).

**Figure 6-29.3 Symbolic Segment Example**

| Segment Type | | | | | | | | | First Character | | | | | | | | | Second Character | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| Segment type (symbolic) | | | Symbol size in bytes (2 bytes) | | | | | | ASCII 1 (31 hex) | | | | | | | | | ASCII 4 (34 hex) | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-29.6 I/O Assembly Data Attribute Format

### 6-29.6.1 Output Assembly Data Attribute Format

There are no new output assemblies defined for Softstart devices.

### 6-29.6.2 Input Assembly Data Attribute Format

**Table 6-29.6 Instance 60: Basic Softstart Input**

| This is the only required input assembly. for the device type SoftStart (15hex) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | At Reference | Reserved | Reserved | Reserved | Reserved | Running1 | Reserved | Faulted/ Trip |

**Table 6-29.7 Instance 61: Extended Softstart Input (see table for functional assignments)**

| This assembly uses some of the optional attributes. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Byte** | **Bit 7** | **Bit 6** | **Bit 5** | **Bit 4** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| 0 | At Reference | Reserved | CntrlfromNet | Ready | Running2 | Running1 | Warning | Faulted/ Trip |

## 6-29.7 Mapping I/O Assembly Data Attribute Components

**Table 6-29.8 Assembly Data Attribute Mapping**

| Data Name | Class Name | Class Number | Instance | Attribute Name | Attribute Number |
|---|---|---|---|---|---|
| Faulted/ Trip | Control Supervisor | 0x29 | 1 | Faulted/ Trip | 10 |
| Running_1 | Control Supervisor | 0x29 | 1 | Running_1 | 7 |
| Running_2 | Control Supervisor | 0x29 | 1 | Running_2 | 8 |
| Ready | Control Supervisor | 0x29 | 1 | Ready | 9 |
| Warning | Control Supervisor | 0x29 | 1 | Warning | 11 |
| Control From Net | Control Supervisor | 0x29 | 1 | CtrlFromNet | 15 |
| At Reference | Softstart | 0x2D | 1 | AtRef | 1 |

## 6-29.8 Defining Device Configuration

Public access to the Control Supervisor Object and the Overload Object must be supported for configuration of Softstart devices. If supported, optional Parameter Objects may be used to access the various configuration attributes in the Control Supervisor Object and the Overload Object.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-30 Human-Machine Interface (HMI)

**Device Type: 18 Hex**

The Human-Machine Interface (HMI) Device type is based on the Communications Adapter device profile (0x0C), section 6-13. The purpose of this device profile is to allow network tools to identify and distinguish HMI devices from other devices on a network. HMI Device type devices are not interchangeable.

## 6-30.1 Object Model

The Object Model in Figure 6-30.1 represents the minimum support in an HMI Device. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-30.1 Objects Present in an HMI Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Optional | Possibly 1 or more |
| Application | Optional | Possibly 1 or more |

The HMI Device profile cannot specify the definition of the Assembly Object or the type of application objects necessary for device operation if they are used. This portion of the device profile must be supplied by the product developer as described in Sections 6-1 through 6-7 inclusive.

Since the HMI device profile does not define any static Assembly object instances, use of Assembly object instances in the “Open - Defined by the device profile” ranges (see Table 5A-5.2 Assembly Instance ID Ranges) for vendor specific purposes is permitted.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-30.1 Object Model for an HMI Device**

## 6-30.2 How Objects Affect Behavior

The objects for this device affect the device's behavior as shown in the table below.

**Table 6-30.2 Object Effect on Behavior**

| Object | Effect on behavior |
|---|---|
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | May be used to define input/output and configuration data formats |
| Application | May be used to define device operation |

## 6-30.3 Defining Object Interfaces

The objects in the HMI Device have the interfaces listed in the following table:

**Table 6-30.3 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| Application | Assembly or Message Router |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-31 Mass Flow Controller Device

**Device Type: 1A Hex**

A Mass Flow Controller is a device that measures and controls the mass flow rate of gas or liquid. It contains three principle components: a mass flow rate sensor which can be one of a variety of types, including thermal or pressure-based; a mass flow rate metering valve which can be actuated by one of a variety of actuator types, including solenoid, voice coil or piezo; and, a controller which closes the loop by receiving a setpoint and driving the actuator such that the mass flow rate is controlled to the setpoint.

## 6-31.1 Object Model

The Object Model in Figure 6-31.1 represents a Mass Flow Controller Device. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-31.1 Objects Present in a Mass Flow Controller Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Required | at least 1 Input and 1 Output |
| S-Device Supervisor | Required | 1 |
| S-Gas Calibration | Optional | 0 or More |
| S-Analog Sensor | Required | 1 |
| S-Analog Actuator | Conditional * | 1 |
| S-Single Stage Controller | Conditional * | 1 |

* Required for a Mass Flow Controller, a device that contains a Valve and a Controller. Not supported in a Mass Flow Meter Device (an MFC without a Valve or a Controller).

### 6-31.1.1 Class Subclasses

Each class level subclass defines a unique meaning for an overlapping range of class attribute IDs and/or class service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given class is identified by the value of its Subclass class attribute. There are no class level subclasses specified for this device.

### 6-31.1.2 Instance Subclasses

Each instance level subclass defines a unique meaning for an overlapping range of instance attribute IDs and/or instance service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given instance is identified by the value of its Subclass instance attribute. The following tables identify which object instance IDs are assigned subclasses for this device.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-31.2 S-Analog Sensor Object Subclasses**

| Instance ID | Subclass Name | Subclass ID (Attribute 99 Value) | Required | Function | Restrictions |
|---|---|---|---|---|---|
| 1 | Flow Diagnostics | 01 | Required | Added diagnostics for MFC | None |

**Table 6-31.3 S-Gas Calibration Object Subclasses**

| Instance ID | Subclass Name | Subclass ID (Attribute 99 Value) | Required | Function | Restrictions |
|---|---|---|---|---|---|
| 1 | Standard T & P | 01 | Optional | Standard Temperature and Pressure | None |

**Figure 6-31.1 Object Model for the MFC Device**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-31.2 How Objects Affect Behavior

**Table 6-31.4 Object Affect on Behavior**

| Object | Effect on behavior |
|---|---|
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines input/output and configuration data format |
| S-Device Supervisor | Supports the Stop, Start, Reset, Abort, Recover and Perform_Diagnostic services for ALL Application Objects in the device and consolidates the Exception Conditions and Application Objects' Status.<br>This object behaves differently from the Identity Object in that the S-Device Supervisor object provides a single point of access to the Application Objects only; it does not effect the CIP specific objects (i.e., Identity, Network Specific Link Object, Connection, etc.). |
| S-Gas Calibration | Modifies the correction algorithm of the S-Analog Sensor object which includes the selection mechanism to enable an S-Gas Calibration object instance. |
| S-Analog Sensor | Feeds the process variable to the Single Stage Controller object |
| S-Single Stage Controller | Feeds the control variable to the Analog Actuator object |
| S-Analog Actuator | Operates the Flow Control Valve of the device |

## 6-31.3 Defining Object Interfaces

**Table 6-31.5 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| S-Device Supervisor | Assembly or Message Router |
| S-Gas Calibration | Message Router |
| S-Analog Sensor | Assembly or Message Router |
| S-Single Stage Controller | Assembly or Message Router |
| S-Analog Actuator | Assembly or Message Router |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-31.4 I/O Assembly Instances

The following table identifies the I/O assembly instances supported by the MFC.

**Table 6-31.6 I/O Assembly Instances**

| Number | Required | Type | Name |
|---|---|---|---|
| 1 | N | Input | Flow |
| 2 | Y (default) | Input | Status and Flow |
| 3 | N | Input | Status, Flow and Valve |
| 4 | N | Input | Status, Flow, and Setpoint |
| 5 | N | Input | Status, Flow, Setpoint and Valve |
| 6 | Y | Input | Status, Flow, Setpoint, Override and Valve |
| 7 | Y (default) | Output | Setpoint |
| 8 | Y | Output | Override and Setpoint |
| 9 | N | Input | Status |
| 10 | N | Input | Exception Detail Alarm |
| 11 | N | Input | Exception Detail Warning |
| 12 | N | Input | Exception Detail Alarm and Exception Detail Warning |
| 13 | N | Input | FP-Flow |
| 14 | Y | Input | FP-Status and Flow |
| 15 | N | Input | FP-Status, Flow and Valve |
| 16 | N | Input | FP-Status, Flow, and Setpoint |
| 17 | N | Input | FP-Status, Flow, Setpoint and Valve |
| 18 | Y | Input | FP-Status, Flow, Setpoint, Override and Valve |
| 19 | Y | Output | FP-Setpoint |
| 20 | Y | Output | FP-Override and Setpoint |

## 6-31.5 I/O Assembly Object Instance Data Attribute Format

The manufacturer of a Mass Flow Controller Device must specify which Assembly instances are supported by the device.

The I/O Assembly DATA attribute has the format shown below.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-31.7 I/O Assembly Data Attribute Format**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | Flow (low byte) | | | | | | | |
| | 1 | Flow (high byte) | | | | | | | |
| 2 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow (high byte) | | | | | | | |
| 3 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow (high byte) | | | | | | | |
| | 3 | Valve (low byte) | | | | | | | |
| | 4 | Valve (high byte) | | | | | | | |
| 4 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow (high byte) | | | | | | | |
| | 3 | Setpoint (low byte) | | | | | | | |
| | 4 | Setpoint (high byte) | | | | | | | |
| 5 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow (high byte) | | | | | | | |
| | 3 | Setpoint (low byte) | | | | | | | |
| | 4 | Setpoint (high byte) | | | | | | | |
| | 5 | Valve (low byte) | | | | | | | |
| | 6 | Valve (high byte) | | | | | | | |
| 6 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow (high byte) | | | | | | | |
| | 3 | Setpoint (low byte) | | | | | | | |
| | 4 | Setpoint (high byte) | | | | | | | |
| | 5 | Override | | | | | | | |
| | 6 | Valve (low byte) | | | | | | | |
| | 7 | Valve (high byte) | | | | | | | |
| 7 | 0 | Setpoint (low byte) | | | | | | | |
| | 1 | Setpoint (high byte) | | | | | | | |
| 8 | 0 | Override | | | | | | | |
| | 1 | Setpoint (low byte) | | | | | | | |
| | 2 | Setpoint (high byte) | | | | | | | |
| 9 | 0 | Status | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Mass Flow Controller Device, Type: $1A_{Hex}$

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 0 | Status | | | | | | | |
| | 1 | Exception Detail Alarm 0 (size, common) | | | | | | | |
| | 2 | Exception Detail Alarm 1 (common 0) | | | | | | | |
| | 3 | Exception Detail Alarm 2 (common 1) | | | | | | | |
| | 4 | Exception Detail Alarm 3 (size, device) | | | | | | | |
| | 5 | Exception Detail Alarm 4 (device 0) | | | | | | | |
| | 6 | Exception Detail Alarm 5 (size, manufacturer) | | | | | | | |
| | 7 | Exception Detail Alarm 6 (manufacturer 0) | | | | | | | |
| 11 | 0 | Status | | | | | | | |
| | 1 | Exception Detail Warning 0 (size, common) | | | | | | | |
| | 2 | Exception Detail Warning 1 (common 0) | | | | | | | |
| | 3 | Exception Detail Warning 2 (common 1) | | | | | | | |
| | 4 | Exception Detail Warning 3 (size, device) | | | | | | | |
| | 5 | Exception Detail Warning 4 (device 0) | | | | | | | |
| | 6 | Exception Detail Warning 5 (size, manufacturer) | | | | | | | |
| | 7 | Exception Detail Warning 6 (manufacturer, 0) | | | | | | | |
| 12 | 0 | Status | | | | | | | |
| | 1 | Exception Detail Alarm 0 (size, common) | | | | | | | |
| | 2 | Exception Detail Alarm 1 (common 0) | | | | | | | |
| | 3 | Exception Detail Alarm 2 (common 1) | | | | | | | |
| | 4 | Exception Detail Alarm 3 (size, device) | | | | | | | |
| | 5 | Exception Detail Alarm 4 (device 0) | | | | | | | |
| | 6 | Exception Detail Alarm 5 (size, manufacturer) | | | | | | | |
| | 7 | Exception Detail Alarm 6 (manufacturer, 0) | | | | | | | |
| | 8 | Exception Detail Warning 0 (size, common) | | | | | | | |
| | 9 | Exception Detail Warning 1 (common 0) | | | | | | | |
| | 10 | Exception Detail Warning 2 (common 1) | | | | | | | |
| | 11 | Exception Detail Warning 3 (size, device) | | | | | | | |
| | 12 | Exception Detail Warning 4 (device 0) | | | | | | | |
| | 13 | Exception Detail Warning 5 (size, manufacturer) | | | | | | | |
| | 14 | Exception Detail Warning 6 (manufacturer, 0) | | | | | | | |
| 13 | 0 | Flow (low byte) | | | | | | | |
| | 1 | Flow | | | | | | | |
| | 2 | Flow | | | | | | | |
| | 3 | Flow (high byte) | | | | | | | |
| 14 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow | | | | | | | |
| | 3 | Flow | | | | | | | |
| | 4 | Flow (high byte) | | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Mass Flow Controller Device, Type: $1A_{Hex}$

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 15 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow | | | | | | | |
| | 3 | Flow | | | | | | | |
| | 4 | Flow (high byte) | | | | | | | |
| | 5 | Valve (low byte) | | | | | | | |
| | 6 | Valve | | | | | | | |
| | 7 | Valve | | | | | | | |
| | 8 | Valve (high byte) | | | | | | | |
| 16 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow | | | | | | | |
| | 3 | Flow | | | | | | | |
| | 4 | Flow (high byte) | | | | | | | |
| | 5 | Setpoint (low byte) | | | | | | | |
| | 6 | Setpoint | | | | | | | |
| | 7 | Setpoint | | | | | | | |
| | 8 | Setpoint (high byte) | | | | | | | |
| 17 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow | | | | | | | |
| | 3 | Flow | | | | | | | |
| | 4 | Flow (high byte) | | | | | | | |
| | 5 | Setpoint (low byte) | | | | | | | |
| | 6 | Setpoint | | | | | | | |
| | 7 | Setpoint | | | | | | | |
| | 8 | Setpoint (high byte) | | | | | | | |
| | 9 | Valve (low byte) | | | | | | | |
| | 10 | Valve | | | | | | | |
| | 11 | Valve | | | | | | | |
| | 12 | Valve (high byte) | | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 18 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow | | | | | | | |
| | 3 | Flow | | | | | | | |
| | 4 | Flow (high byte) | | | | | | | |
| | 5 | Setpoint (low byte) | | | | | | | |
| | 6 | Setpoint | | | | | | | |
| | 7 | Setpoint | | | | | | | |
| | 8 | Setpoint (high byte) | | | | | | | |
| | 9 | Override | | | | | | | |
| | 10 | Valve (low byte) | | | | | | | |
| | 11 | Valve | | | | | | | |
| | 12 | Valve | | | | | | | |
| | 13 | Valve (high byte) | | | | | | | |
| 19 | 0 | Setpoint (low byte) | | | | | | | |
| | 1 | Setpoint | | | | | | | |
| | 2 | Setpoint | | | | | | | |
| | 3 | Setpoint (high byte) | | | | | | | |
| 20 | 0 | Override | | | | | | | |
| | 1 | Setpoint (low byte) | | | | | | | |
| | 2 | Setpoint | | | | | | | |
| | 3 | Setpoint | | | | | | | |
| | 4 | Setpoint (high byte) | | | | | | | |

### 6-31.6 Mapping I/O Assembly Data Attribute Components

Each of the *S-Analog Sensor*, *S-Analog Actuator* and *S-Single Stage Controller* object definitions specifies a behavior that modifies the *Data Type* of certain attributes based upon the first valid I/O connection established to an Assembly Object instance. In order to maintain consistency, this device type will only allow connections to either INT or REAL based Assembly instances. Once a valid connection is established, attempts to configure connections to a different type of Assembly instance will return an error.

The following table indicates the I/O assembly Data attribute mapping for this MFC device.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-31.8 I/O Assembly Data Mapping**

| Data Component | Class | | Instance | Attribute | | |
|---|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number | Type |
| Flow | S-Analog Sensor | 31hex | 1 | Indicated Flow | 6 | INT |
| Valve | S-Analog Actuator | 32hex | 1 | Value | 6 | INT |
| Override | S-Analog Actuator | 32hex | 1 | Override | 5 | USINT |
| Setpoint | S-Single Stage Controller | 33hex | 1 | Setpoint | 6 | INT |
| Status | S-Device Supervisor | 30hex | 1 | Exception Status | 12 | BYTE |
| Exception Detail Alarm | S-Device Supervisor | 30hex | 1 | Exception Detail Alarm | 13 | STRUCT |
| Exception Detail Warning | S-Device Supervisor | 30hex | 1 | Exception Detail Warning | 14 | STRUCT |
| FP-Flow | S-Analog Sensor | 31hex | 1 | Indicated Flow | 6 | REAL |
| FP-Valve | S-Analog Actuator | 32hex | 1 | Value | 6 | REAL |
| FP-Setpoint | S-Single Stage Controller | 33hex | 1 | Setpoint | 6 | REAL |

## 6-31.7 Object Limitations and Specific Definitions

This section describes limitations and specific definitions applicable to the listed objects of this device profile when these objects are applied in this type of device.

### 6-31.7.1 S-Device Supervisor Object Instance

#### 6-31.7.1.1 Limitations

**Table 6-31.9 Device Type Attribute Limitations**

| Attribute | Limitation |
|---|---|
| Device Type | Supported Values:<br>“MFC” = Mass Flow Controller device<br>“MFM” = Mass Flow Meter device |

#### 6-31.7.1.2 Specific Definition

The following table specifies the data attribute bit mapping for the **Device Exception Detail** bytes for this MFC device. For more descriptive information, see the definition of the S-Device Supervisor Object Class. Noted, for each entry, is the Object from which the Status byte/bit is mapped. See the object specification for the detailed bit mapping.

Any Exception Bit not supported must default to 0. Note that this profile allows for only one byte of manufacturer specific exception detail.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-31.10 Device Exception Detail Attribute Mapping**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| MFC Device Exception Detail Size | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| MFC Device Exception Detail | Reserved 0 | Reserved 0 | Valve High S-Analog Actuator | Valve Low S-Analog Actuator | Flow Control S-Single Stage Controller | Flow High S-Analog Sensor | Flow Low S-Analog Sensor | Reading Valid * S-Analog Sensor |
| Manufacturer Exception Detail Size | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Manufacturer Exception Detail | 8 Bits defined by Manufacturer | | | | | | | |
| * Only used in the Warning Exception Detail, this bit is always = 0 in the Alarm Exception Detail. | | | | | | | | |

## 6-31.7.2 S-Analog Sensor Object Instance

### 6-31.7.2.1 Limitations

**Table 6-31.11 Attribute Limitations**

| Attribute | Limitation | Requirement | Default |
|---|---|---|---|
| Data Type | Supported Values = {INT; REAL} | Supported Values = {INT; REAL} | INT |
| Data Units | Supported Values = {Counts & Units of the Flow Group} (see Appendix D) | Supported Values = {Counts; sccm} | Counts |
| Offset-A Data Type | Supported Values = {INT; REAL} | Supported Values = {INT; REAL} | INT |
| Gain Data Type | Supported Values = {INT; REAL} | Supported Values = {INT; REAL} | INT |

## 6-31.7.3 S-Analog Actuator Object Instance

### 6-31.7.3.1 Limitations

**Table 6-31.12 Attribute Limitations**

| Attribute | Limitation | Requirement | Default |
|---|---|---|---|
| Data Type | Supported Values = {INT; REAL} | Supported Values = {INT; REAL} | INT |
| Data Units | Supported Values = {Counts, %, Voltage, Current & Units of the Flow Group} (see Appendix D) | Supported Values = {Counts; %} | Counts |
| Gain Data Type | Supported Values = {INT; REAL} | Supported Values = {INT; REAL} | INT |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-31.7.4 S-Single Stage Controller Object Instance

#### 6-31.7.4.1 Limitations

**Table 6-31.13 Attribute Limitations**

| Attribute | Limitation | Requirement | Default |
|---|---|---|---|
| Data Type | Supported Values = {INT; REAL} | Supported Values = {INT; REAL} | INT |
| Data Units | Supported Values = {Counts, %, Voltage, Current & Units of the Flow Group} (see Appendix D) | Supported Values = {Counts; %} | Counts |
| Process Variable | Not accessible over the network. The Process Variable input to this object instance is the value of the S-Analog Sensor object instance Value attribute. | N.A. | N.A. |
| CV Data Type | Not supported | N.A. | N.A. |
| Control Variable | Not accessible over the network, The Control Variable output from this object is the value of the S-Analog Actuator object instance Value attribute. | N.A. | N.A. |

### 6-31.8 Defining Device Configuration

Public access to the S-Device Supervisor, S-Analog Sensor, S-Analog Actuator, S-Single Stage Controller, and S-Gas Calibration Objects by the Message Router must be supported for configuration of this device type.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-32 Vacuum/Pressure Gauge Device

**Device Type: 1CHex**

The objective of this profile is to provide a vacuum or pressure measurement profile, which is inclusive of all technologies used to provide the pressure reading. By use of ”gauge” subclasses of the S-Analog Sensor, this profile can apply to Heat Transfer Gauges (Convection, Pirani, Thermocouple), Hot Cathode Ion Gauge, Cold Cathode Ion Gauge or a Diaphragm Gauge. The gauge subclasses provide calibration, control and status attributes unique to each gauge type. The S-Analog Sensor Object provides a pressure value to the Assembly instances delineated in this profile.

Combination and Multiple Gauges

This profile has been structured to facilitate use of ”Combination” gauges - multiple gauges each covering separate, contiguous ranges of pressure, only one gauge active and providing one Pressure Value at any particular time; and ”Multiple” gauges – in which all gauge readings are simultaneously available (but not necessarily valid). In both cases, multiple instances of the S-Analog Sensor Object are used.

## 6-32.1 Object Model

The Object Model in Figure 6-32.1 represents a Vacuum/Pressure Gauge Device.

The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-32.1 Objects Present in a Vacuum/Pressure Gauge Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Required | At least 1 I/O and 1 Explicit |
| S-Device Supervisor | Required | 1 |
| S-Analog Sensor | 1 Subclass required as a minimum<br>See S-Analog Sensor Subclasses in object model above. | 1 or more |
| S-Gas Calibration | Optional | 1 or more |
| Trip Point | Optional | 1 … 8 |
| Discrete Output Point | Optional | 1 or more |
| Analog Output Point | Optional | 1 or more |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-32.2 Class Subclasses

Each class level subclass defines a unique meaning for an overlapping range of class attribute IDs and/or class service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given class is identified by the value of its Subclass class attribute.

**Table 6-32.2 S-Analog Sensor Class Level Object Subclasses**

| Instance ID | Subclass Name | Subclass ID (Attribute 99 Value) | Required | Function | Restrictions |
|---|---|---|---|---|---|
| Class | Instance Selector | 01 | Conditional* | Data Flow from the Active Instance | None |

* Required for combination gauges only

## 6-32.3 Instance Subclasses

Each instance level subclass defines a unique meaning for an overlapping range of instance attribute IDs and/or instance service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given instance is identified by the value of its Subclass instance attribute. The following tables identify which object instance IDs are assigned subclasses for this device.

**Table 6-32.3 S-Analog Sensor Object Instance Level Subclasses**

| Instance ID | Subclass Name | Subclass ID (Attribute 99 Value) | Required | Function | Restrictions |
|---|---|---|---|---|---|
| * | Heat Transfer Vacuum Gauge | 02 | Conditional ** | Gauge Application | None |
| * | Diaphragm Gauge Gauge | 03 | Conditional ** | Gauge Application | None |
| * | Cold Cathode Ion Gauge | 04 | Conditional ** | Gauge Application | None |
| * | Hot Cathode Ion Gauge | 05 | Conditional ** | Gauge Application | None |

* Instance IDs are vendor specific
** The Gauge type Subclass is required if the referenced gauge type is implemented.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-32.1 Object Model**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-32.4 How Objects Affect Behavior

**Table 6-32.4 Object Effect on Behavior**

| Object | Effect on behavior |
|---|---|
| Identity | Supports the Reset service. Upon receipt of a *Reset* Service Request of any *Type*, the Identity Object sends a *Reset* Service Request to the S-Device Supervisor. |
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines input/output and configuration data format |
| S-Device Supervisor | Supports the Stop, Start, Reset, Abort, Recover and Perform_Diagnostic services for ALL Application Objects in the device and consolidates the Exception Conditions and Application Objects' Status.<br>This object behaves differently from the Identity Object in that the S-Device Supervisor object provides a single point of access to the Application Objects only; it does not effect the DeviceNet objects (i.e., Identity, DeviceNet, Connection, etc.). |
| S-Analog Sensor | Each instance of this object provides a calibrated pressure value from a pressure transducer. This object can also be used to supply an internal potentiometer position used for calibration purposes. Each instance will most likely use a gauge subclass; which subclass is used is determined by the gauge technology used. |
| S-Gas Calibration | Modifies the correction algorithm of the S-Analog Sensor object. The Gas Calibration Instance attribute of that object specifies which instance is active. |
| Trip Point | Provides a process trip point comparator for the S-Analog Sensor value. Each instance is linked to an S-Analog Sensor instance. The output of this object may be used to drive the Discrete Output Point object. |
| Discrete Output Point | Reflects status of Trip Point object instances. |
| Analog Output Point | Provides an Analog Output from the device which is fed from the S-Analog Sensor value. This analog value may be of a data type and data units different from those of the S-Analog Sensor. This object is required only if the output value is supported as visible to the network. |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-32.5 Defining Object Interfaces

**Table 6-32.5 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| S-Device Supervisor | Assembly or Message Router |
| S-Analog Sensor | Assembly or Message Router |
| S-Gas Calibration | Message Router |
| Trip Point | Assembly or Message Router |
| Discrete Output Point | Message Router |
| Analog Output Point | Message Router |

## 6-32.6 I/O Assembly Instances

The manufacturer of a Gauge Device must specify which Assembly instances are supported by the device.

The *S-Analog Sensor* object definition specifies a behavior that modifies the *Data Type* of certain attributes based upon the first valid I/O connection established to an Assembly Object instance. In order to maintain consistency, this device type will only allow connections to either INT or REAL based Assembly instances. Once a valid connection is established, attempts to configure connections, or otherwise access data, to a different type of Assembly instance will return a RESOURCE UNAVAILABLE error.

### Combination Gauges

Though the device supports multiple instances of the S-Analog Sensor class, only a single *Pressure Value* is produced in the Assemblies. The S-Analog Sensor class-level attribute *Active Instance Number* identifies the object instance that is currently active and providing its *Pressure Value* to the *Active Pressure Value* which is, in turn, produced by the input assemblies. Note that this behavior does not apply to the Trip Point and the Analog Output Point objects, which are directly linked to S-Analog Sensor instances.

### Multiple Gauges

Multiple pressure values are available from multiple S-Analog Sensor instances and are provided in each assembly instance that contains multiple *Pressure Value* members indicated by *Pressure Value n,* where *n* corresponds to the S-Analog Sensor instance ID. The maximum number for *n* is specified by the value of the S-Analog Sensor object class attribute *Number of Gauges*. For devices with fewer gauges than the number specified in a given input assembly, the missing Data Components of that assembly will be set to zero (0). Note that for the purpose of bandwidth optimization, I/O connections to such assemblies where less than the included number of instances are valid, the I/O connection produce length can be set accordingly.

The following table identifies the I/O assembly instances. The manufacturer must specify which Assembly instances are supported by the device.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-32.6 I/O Assembly Instances**

| Instance | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | N | Input | 0 - 1 | INT Pressure Value | | | | | | | |
| 2 | Y (default) | Input | 0 | Exception Status | | | | | | | |
| | | | 1 - 2 | INT Pressure Value | | | | | | | |
| 3 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Trip Status Inst. 8 | Trip Status Inst. 7 | Trip Status Inst. 6 | Trip Status Inst. 5 | Trip Status Inst. 4 | Trip Status Inst. 3 | Trip Status Inst. 2 | Trip Status Inst. 1 |
| | | | 2 - 3 | INT Pressure Value | | | | | | | |
| 4 | N | Input | 0 - 3 | REAL Pressure Value | | | | | | | |
| 5 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 - 4 | REAL Pressure Value | | | | | | | |
| 6 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Trip Status Inst. 8 | Trip Status Inst. 7 | Trip Status Inst. 6 | Trip Status Inst. 5 | Trip Status Inst. 4 | Trip Status Inst. 3 | Trip Status Inst. 2 | Trip Status Inst. 1 |
| | | | 2 - 5 | REAL Pressure Value | | | | | | | |
| 7 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Trip Status Inst. 8 | Trip Status Inst. 7 | Trip Status Inst. 6 | Trip Status Inst. 5 | Trip Status Inst. 4 | Trip Status Inst. 3 | Trip Status Inst. 2 | Trip Status Inst. 1 |
| 8 | N | Input | 0 | Exception Status | | | | | | | |
| 9 | N | Input | 0 - 1 | Active Instance | | | | | | | |
| | | | 2 - 3 | INT Active Pressure Value | | | | | | | |
| 10 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 - 2 | Active Instance | | | | | | | |
| | | | 3 - 4 | INT Active Pressure Value | | | | | | | |
| 11 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Trip Status Inst. 8 | Trip Status Inst. 7 | Trip Status Inst. 6 | Trip Status Inst. 5 | Trip Status Inst. 4 | Trip Status Inst. 3 | Trip Status Inst. 2 | Trip Status Inst. 1 |
| | | | 2 - 3 | Active Instance | | | | | | | |
| | | | 4 - 5 | INT Active Pressure Value | | | | | | | |
| 12 | N | Input | 0 - 1 | Active Instance | | | | | | | |
| | | | 2 - 5 | REAL Active Pressure Value | | | | | | | |
| 13 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 - 2 | Active Instance | | | | | | | |
| | | | 3 - 6 | REAL Active Pressure Value | | | | | | | |
| 14 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Trip Status Inst. 8 | Trip Status Inst. 7 | Trip Status Inst. 6 | Trip Status Inst. 5 | Trip Status Inst. 4 | Trip Status Inst. 3 | Trip Status Inst. 2 | Trip Status Inst. 1 |
| | | | 2 - 3 | Active Instance | | | | | | | |
| | | | 4 - 7 | REAL Active Pressure Value | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Vacuum/Pressure Gauge Device, Type: $1C_{Hex}$

| Instance | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | N | Input | 0 - 1 | INT Pressure Value 1 | | | | | | | |
| | | | 2 - 3 | INT Pressure Value 2 | | | | | | | |
| | | | 4 - 5 | INT Pressure Value 3 | | | | | | | |
| | | | 6 - 7 | INT Pressure Value 4 | | | | | | | |
| 16 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 - 2 | INT Pressure Value 1 | | | | | | | |
| | | | 3 - 4 | INT Pressure Value 2 | | | | | | | |
| | | | 5 - 6 | INT Pressure Value 3 | | | | | | | |
| | | | 7 - 8 | INT Pressure Value 4 | | | | | | | |
| 17 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Trip Status Inst. 8 | Trip Status Inst. 7 | Trip Status Inst. 6 | Trip Status Inst. 5 | Trip Status Inst. 4 | Trip Status Inst. 3 | Trip Status Inst. 2 | Trip Status Inst. 1 |
| | | | 2 - 3 | INT Pressure Value 1 | | | | | | | |
| | | | 4 - 5 | INT Pressure Value 2 | | | | | | | |
| | | | 6 - 7 | INT Pressure Value 3 | | | | | | | |
| | | | 8 - 9 | INT Pressure Value 4 | | | | | | | |
| 18 | N | Input | 0 - 3 | REAL Pressure Value 1 | | | | | | | |
| | | | 4 - 7 | REAL Pressure Value 2 | | | | | | | |
| | | | 8 - 11 | REAL Pressure Value 3 | | | | | | | |
| | | | 12 - 15 | REAL Pressure Value 4 | | | | | | | |
| 19 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 - 4 | REAL Pressure Value 1 | | | | | | | |
| | | | 5 - 8 | REAL Pressure Value 2 | | | | | | | |
| | | | 9 - 12 | REAL Pressure Value 3 | | | | | | | |
| | | | 13 - 16 | REAL Pressure Value 4 | | | | | | | |
| 20 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Trip Status Inst. 8 | Trip Status Inst. 7 | Trip Status Inst. 6 | Trip Status Inst. 5 | Trip Status Inst. 4 | Trip Status Inst. 3 | Trip Status Inst. 2 | Trip Status Inst. 1 |
| | | | 2 - 5 | REAL Pressure Value 1 | | | | | | | |
| | | | 6 - 9 | REAL Pressure Value 2 | | | | | | | |
| | | | 10 - 13 | REAL Pressure Value 3 | | | | | | | |
| | | | 14 - 17 | REAL Pressure Value 4 | | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Vacuum/Pressure Gauge Device, Type: $1C_{Hex}$

| Instance | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 21 | N | Input | 0 - 1 | INT Pressure Value 1 | | | | | | | |
| | | | 2 - 3 | INT Pressure Value 2 | | | | | | | |
| | | | 4 - 5 | INT Pressure Value 3 | | | | | | | |
| | | | 6 - 7 | INT Pressure Value 4 | | | | | | | |
| | | | 8 - 9 | INT Pressure Value 5 | | | | | | | |
| | | | 10 - 11 | INT Pressure Value 6 | | | | | | | |
| | | | 12 - 13 | INT Pressure Value 7 | | | | | | | |
| | | | 14 - 15 | INT Pressure Value 8 | | | | | | | |
| 22 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 - 2 | INT Pressure Value 1 | | | | | | | |
| | | | 3 - 4 | INT Pressure Value 2 | | | | | | | |
| | | | 5 - 6 | INT Pressure Value 3 | | | | | | | |
| | | | 7 - 8 | INT Pressure Value 4 | | | | | | | |
| | | | 9 - 10 | INT Pressure Value 5 | | | | | | | |
| | | | 11 - 12 | INT Pressure Value 6 | | | | | | | |
| | | | 13 - 14 | INT Pressure Value 7 | | | | | | | |
| | | | 15 - 16 | INT Pressure Value 8 | | | | | | | |
| 23 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Trip Status Inst. 8 | Trip Status Inst. 7 | Trip Status Inst. 6 | Trip Status Inst. 5 | Trip Status Inst. 4 | Trip Status Inst. 3 | Trip Status Inst. 2 | Trip Status Inst. 1 |
| | | | 2 - 3 | INT Pressure Value 1 | | | | | | | |
| | | | 4 - 5 | INT Pressure Value 2 | | | | | | | |
| | | | 6 - 7 | INT Pressure Value 3 | | | | | | | |
| | | | 8 - 9 | INT Pressure Value 4 | | | | | | | |
| | | | 10 - 11 | INT Pressure Value 5 | | | | | | | |
| | | | 12 - 13 | INT Pressure Value 6 | | | | | | | |
| | | | 14 - 15 | INT Pressure Value 7 | | | | | | | |
| | | | 16 - 17 | INT Pressure Value 8 | | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Vacuum/Pressure Gauge Device, Type: $1C_{Hex}$

| Instance | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 24 | N | Input | 0 - 3 | REAL Pressure Value 1 | | | | | | | |
| | | | 4 - 7 | REAL Pressure Value 2 | | | | | | | |
| | | | 8 - 11 | REAL Pressure Value 3 | | | | | | | |
| | | | 12 - 15 | REAL Pressure Value 4 | | | | | | | |
| | | | 16 - 19 | REAL Pressure Value 5 | | | | | | | |
| | | | 20 - 23 | REAL Pressure Value 6 | | | | | | | |
| | | | 24 - 27 | REAL Pressure Value 7 | | | | | | | |
| | | | 28 - 31 | REAL Pressure Value 8 | | | | | | | |
| 25 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 - 4 | REAL Pressure Value 1 | | | | | | | |
| | | | 5 - 8 | REAL Pressure Value 2 | | | | | | | |
| | | | 9 - 12 | REAL Pressure Value 3 | | | | | | | |
| | | | 13 - 16 | REAL Pressure Value 4 | | | | | | | |
| | | | 17 - 20 | REAL Pressure Value 5 | | | | | | | |
| | | | 21 - 24 | REAL Pressure Value 6 | | | | | | | |
| | | | 25 - 28 | REAL Pressure Value 7 | | | | | | | |
| | | | 29 - 32 | REAL Pressure Value 8 | | | | | | | |
| 26 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Trip Status Inst. 8 | Trip Status Inst. 7 | Trip Status Inst. 6 | Trip Status Inst. 5 | Trip Status Inst. 4 | Trip Status Inst. 3 | Trip Status Inst. 2 | Trip Status Inst. 1 |
| | | | 2 - 5 | REAL Pressure Value 1 | | | | | | | |
| | | | 6 - 9 | REAL Pressure Value 2 | | | | | | | |
| | | | 10 - 13 | REAL Pressure Value 3 | | | | | | | |
| | | | 14 - 17 | REAL Pressure Value 4 | | | | | | | |
| | | | 18 - 21 | REAL Pressure Value 5 | | | | | | | |
| | | | 22 - 25 | REAL Pressure Value 6 | | | | | | | |
| | | | 26 - 29 | REAL Pressure Value 7 | | | | | | | |
| | | | 30 - 33 | REAL Pressure Value 8 | | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-32.7 Output Assemblies for Hot Cathode Ion Gauges**

| Inst ID | Req'd | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 200 | N | Output | 0 | Active Degas Filament 8 | Active Degas Filament 7 | Active Degas Filament 6 | Active Degas Filament 5 | Active Degas Filament 4 | Active Degas Filament 3 | Active Degas Filament 2 | Active Degas Filament 1 |
| | | | 1 | Active Filament 8 | Active Filament 7 | Active Filament 6 | Active Filament 5 | Active Filament 4 | Active Filament 3 | Active Filament 2 | Active Filament 1 |
| | | | 2 | reserved | reserved | reserved | reserved | reserved | Clear Emission Off Alarm | Set Degas ON/ OFF | Set Emission ON/ OFF |
| 201 | N | Output | 0 | Active Filament 8 | Active Filament 7 | Active Filament 6 | Active Filament 5 | Active Filament 4 | Active Filament 3 | Active Filament 2 | Active Filament 1 |
| | | | 1 | reserved | reserved | reserved | reserved | reserved | Clear Emission Off Alarm | Set Degas State ON/OFF | Set Emission State ON/OFF |
| 202 | N | Output | 0 | Active Degas Filament 8 | Active Degas Filament 7 | Active Degas Filament 6 | Active Degas Filament 5 | Active Degas Filament 4 | Active Degas Filament 3 | Active Degas Filament 2 | Active Degas Filament 1 |
| | | | 1 | Active Filament 8 | Active Filament 7 | Active Filament 6 | Active Filament 5 | Active Filament 4 | Active Filament 3 | Active Filament 2 | Active Filament 1 |
| | | | 2-5 | Emission Current | | | | | | | |
| | | | 6 | reserved | reserved | reserved | reserved | reserved | Clear Emission Off Alarm | Set Degas State ON/OFF | Set Emission State ON/OFF |
| 203 | N | Output | 0 | Active Filament 8 | Active Filament 7 | Active Filament 6 | Active Filament 5 | Active Filament 4 | Active Filament 3 | Active Filament 2 | Active Filament 1 |
| | | | 1-4 | Emission Current | | | | | | | |
| | | | 5 | reserved | reserved | reserved | reserved | reserved | Clear Emission Off Alarm | Set Degas State ON/OFF | Set Emission State ON/OFF |

If a fault condition occurs (for example overpressure emission off), the state of the ion gauge will be off and the fault condition must be cleared (Clear Emission Off Alarm) before pulling the "Set Emission State" and/or "Set Degas State" bits to zero then one again to light the ion gauge.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-32.8 Output assemblies for Cold Cathode Ion Gauges**

| Inst ID | Req'd | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 200 | N | Output | 0 | reserved | reserved | reserved | reserved | reserved | Clear Overpressure High Voltage Off Alarm | reserved | Set High Voltage State ON/OFF |
| 201 | N | Output | 0-3 | High Voltage | | | | | | | |
| | | | 4 | reserved | reserved | reserved | reserved | reserved | Clear Overpressure High Voltage Off Alarm | reserved | Set High Voltage State ON/OFF |
| 202 | N | Output | 0-3 | High Voltage | | | | | | | |
| | | | 4-7 | Sensitivity | | | | | | | |
| | | | 8 | reserved | reserved | reserved | reserved | reserved | Clear Overpressure High Voltage Off Alarm | reserved | Set High Voltage State ON/OFF |

If an fault condition occurs (for example overpressure emission off), the state of the ion gauge will be off and the fault condition must be cleared (Clear Overpressure Emission Off Alarm) before pulling the "Set High Voltage State" bit to "zero" then "one" again to light the ion gauge.

All of the elemental components listed above follow standard CIP Data Encoding as specified in Appendix C.

### 6-32.6.1 Mapping I/O Assembly Data Attribute Components

The *Data Type* of the Pressure Value attribute below (and the *Data Type* of other objects used in this profile) is based upon the first valid I/O connection established to an Assembly Object instance.

The following table indicates the I/O assembly Data attribute mapping for this Gauge device.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-32.9 I/O Assembly Data Mapping**

| Data Component Name | Class | Class Number | | Instance Number | Attribute | | |
|---|---|---|---|---|---|---|---|
| | | | Sub-class | | Name | Number | Type |
| **Pressure Value** | S-Analog Sensor | 31 hex | - | 1 – N | Value | 6 | INT or REAL |
| **Trip Status** | Trip Point | 35 hex | - | 1 - N | Status | 7 | BOOL |
| **Exception Status** | S-Device Supervisor | 30 hex | - | 1 | Exception Status | 12 | BYTE |
| **Active Instance** | S-Analog Sensor | 31 hex | - | Class Level | Active Instance Number | 95 | UINT |
| **Active Pressure Value *** | S-Analog Sensor | 31 hex | - | Class Level | Active Value | 94 | INT or REAL |
| **Active Degas Filament** | S-Analog Sensor | 31 hex | 5 | 1 - N | Active Degas Filament | 81 | BYTE |
| **Active Filament** | S-Analog Sensor | 31 hex | 5 | 1 - N | Active Filament | 89 | BYTE |
| **High Voltage** | S-Analog Sensor | 31 hex | 4 | 1 - N | High Voltage | 91 | REAL |
| **Sensitivity** | S-Analog Sensor | 31 hex | 4 | 1 - N | Sensitivity | 92 | REAL |

| | | Class Number | | | Service setting/resetting these bits to 0 or 1 in the I/O assembly is equivalent to performing the service with the value 0 or 1 | | |
|---|---|---|---|---|---|---|---|
| **Data Component Name** | **Class** | | **Sub-class** | **Instance Number** | **Service Name** | **Service Code** | **Type** |
| Set Degas State On / Off | S-Analog Sensor | 31 hex | 5 | 1 - N | Set Degas State | 61 hex | BOOL |
| Set Emission State On / Off | S-Analog Sensor | 31 hex | 5 | 1 - N | Set Emission State | 62 hex | BOOL |
| Clear Emission Off Alarm | S-Analog Sensor | 31 hex | 5 | 1 - N | Clear Emission Off Alarm | 63 hex | BOOL |
| Set High Voltage State On / Off | S-Analog Sensor | 31 hex | 4 | 1 - N | Set High Voltage State | 62 hex | BOOL |
| Clear Overpressure High Voltage Off Alarm | S-Analog Sensor | 31 hex | 4 | 1 - N | Clear Overpressure High Voltage Off Alarm | 63 hex | BOOL |

* For Combination Gauges only

## 6-32.7 Object Limitations and Specific Definitions

### 6-32.7.1 S-Device Supervisor Object Instance

#### 6-32.7.1.1 Exception Attributes Definition

The following table specifies the data attribute bit mapping for the Device Exception Detail bytes for this device. For more descriptive information, see the definition of the S-Device Supervisor Object Class.

All status and ”Sensor” bits originate from the S-Analog Sensor object. The meaning of Sensor Alarm and Sensor Warning is defined by the Gauge Subclass used. See the object specification for the detailed bit mapping. Noted with each entry is the Instance from which the status byte/bit is mapped.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Combination and multiple gauges will have a set of Device Exception details for each gauge. The S-Analog Sensor class attribute *Number of Gauges* is used to determine gauge count.

Any Exception Bit not supported shall be set to zero (0). Note that this profile allows for only one byte of manufacturer exception detail.

**Table 6-32.10 Exception Detail Warning (Attribute 14) for single gauges**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Device Exception Detail Warning Size | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Device 1 Exception Detail Warning Byte 0 | S-Analog Sensor object Instance 1 — Status Extension | | | | | | | |
| Device 1 Exception Detail Warning Byte 1 | S-Analog Sensor object Instance 1 — Sensor Warning — Byte 0 | | | | | | | |
| Device 1 Exception Detail Warning Byte 2 | S-Analog Sensor object Instance 1 — Sensor Warning — Byte 1 | | | | | | | |
| | | | | | | | | |
| Manufacturer Exception Detail Warning Size | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Manufacturer Exception Detail Warning | 8 bits defined by Manufacturer | | | | | | | |

**Table 6-32.11 Exception Detail Warning (Attribute 14) for combination and multigauges:**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Device Exception Detail Warning Size | 0 | 0 | 0 | X | X | X | X | X |
| Device 1 Exception Detail Warning Byte 0 | S-Analog Sensor object Class Level Status Extension | | | | | | | |
| Device 1 Exception Detail Warning Byte 1 | S-Analog Sensor object Instance 1 — Sensor Warning — Byte 0 | | | | | | | |
| Device 1 Exception Detail Warning Byte 2 | S-Analog Sensor object Instance 1 — Sensor Warning — Byte 1 | | | | | | | |
| | | | | | | | | |
| Device N Exception Detail Warning Byte 0 | S-Analog Sensor object Instance N — Sensor Warning — Byte 0 | | | | | | | |
| Device N Exception Detail Warning Byte 1 | S-Analog Sensor object Instance N — Sensor Warning — Byte 1 | | | | | | | |
| | | | | | | | | |
| Manufacturer Exception Detail Warning Size | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Manufacturer Exception Detail Warning | 8 bits defined by Manufacturer | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-32.12 Exception Detail Alarms (Attribute 13)**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Device Exception Detail (Alarm) Size | 0 | 0 | 0 | 0 | X | X | X | X |
| Device 1 Exception Detail (Alarm) Byte 0 | S-Analog Sensor object Instance 1 — Sensor Alarm — Byte 0 | | | | | | | |
| Device 1 Exception Detail (Alarm) Byte 1 | S-Analog Sensor object Instance 1 — Sensor Alarm — Byte 1 | | | | | | | |
| Combination and Multi-Gauge Devices Only: | | | | | | | | |
| Device N Exception Detail (Alarm) Byte 0 | S-Analog Sensor object Instance N — Sensor Alarm — Byte 0 | | | | | | | |
| Device N Exception Detail (Alarm) Byte 1 | S-Analog Sensor object Instance N — Sensor Alarm — Byte 1 | | | | | | | |
| [End Combination and Multi-Gauge Devices Only] | | | | | | | | |
| Manufacturer Exception Detail (Alarm) Size | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Manufacturer Exception Detail (Alarm) | 8 bits defined by Manufacturer | | | | | | | |

### 6-32.7.1.2 Manufacturer's Device Type Attribute Definition

The following limitations apply:

**Table 6-32.13 Device Type Attribute Limitations**

| Attribute ID | Need in implementation | Access Rule | Name | Data Type | Description of Attribute |
|---|---|---|---|---|---|
| 3 | Required | Get | Device Type | SHORT STRING | Supported values:<br>"VG" for single-instance vacuum gauge<br>"CG" for combination gauge<br>"MG" for multiple gauge<br>(ref S-Analog Sensor instance attribute *Subclass Number* to determine specific gauge type) |

### 6-32.7.1.3 Behavior

The behavior of all Pressure/Vacuum Gauge objects are defined by the S-Device Supervisor Object in Chapter 5.

Ion Gauges only – the Emission OFF state will result in the following behavior:

a) S-Analog Sensor Value attribute will revert to the Safe State / Safe Value, if supported. This mimics the non-executing state of the device.
b) The Trip Point (if supported) Status attribute will be set to the unasserted state (Override, if supported, = 1).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-32.7.2 S-Analog Sensor Object Instance

#### 6-32.7.2.1 Limitations

##### 6-32.7.2.1.1 Object Instance Number Assignment

The following instance numbering limitations apply:

**Table 6-32.14 Object Instance Number Limitations**

| Instance | Description |
|---|---|
| 1 ... 20 | Reserved for Pressure Gauges |
| 21 ... 60 | Reserved for Interlock-Relay Setting Monitors |

##### 6-32.7.2.1.2 Data Type, Data Units and Produce Trigger Delta Type

The following attribute limitations apply:

**Table 6-32.15 Data Limitations**

| Attribute | Additions / Limitation | Requirements |
|---|---|---|
| Data Type | Supported Values = {INT, REAL}<br>Access Rule = Get only, after an assembly instance connection is established | Supported Value = {INT} |
| Data Units | Supported Values limited to those found in ”Group 4 – Pressure” of Data Units Appendix D, and *Counts*. | Supported Value = {Counts} |
| Produce Trigger Delta Type | Percent | Percent |

All devices are required to support INT Counts. The meaning of Counts is vendor-specific. Which combinations of Data Units and Data Type are supported are vendor-specific. INT Counts and/or REAL Pressure Units are the most likely options. Other combinations (ex. REAL Counts) may not be supported.

Multiple gauges and combination gauges will require multiple instances of the S-Analog Sensor Object, all of which will support the same Data Type and Data Units.

##### 6-32.7.2.1.3 Alarm and Warning Hysteresis (attributes 19 and 23)

For gauges with a logarithmic vacuum reading (hot cathode ion gauge, cold cathode ion gauge, and heat transfer gauges) the Alarm and Warning Hysteresis attributes determine the amount (in the unit "percent of the associated alarm/warning value" ) by which the Value must recover to clear an Alarm Condition.

For gauges with a linear vacuum reading (Diaphragm Gauge) the Alarm and Warning Hysteresis attributes determine the amount (absolute) by which the Value must recover to clear an Alarm Condition.

#### 6-32.7.2.2 Object Extensions

This profile uses the extensions for the following instance-level subclasses:

**Table 6-32.16 Object Extensions for Instance-level Subclasses**

| Gauge Type | Subclass Number |
|---|---|
| Heat Transfer Vacuum Gauge | 02 |
| Diaphragm Gauge | 03 |
| Cold Cathode Ion Gauge | 04 |
| Hot Cathode Ion Gauge | 05 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-32.7.3 Trip Point Object Instance

#### 6-32.7.3.1 Limitations

##### 6-32.7.3.1.1 Hysteresis (attribute 10)

For gauges with a logarithmic vacuum reading (hot cathode ion gauge, cold cathode ion gauge, and heat transfer gauges) the Hysteresis attribute determines the amount (in the unit "percent of the associated low/high trip point" ) by which the Value must recover to clear a trip point condition.

For gauges with a linear vacuum reading (Diaphragm Gauge) the Hysteresis attribute determines the amount (absolute) by which the Value must recover to clear a trip point condition.

##### 6-32.7.3.1.2 Source (attribute 14)

Abbreviated Packed EPATH — the *Source* attribute in this device type is abbreviated as follows:

Logical Segment, Instance Only, 8-bit Logical Address (see Appendix C).

The following defaults apply:

Class ID = 31hex [S-Analog Sensor].
Attribute ID = 06 [Value].

Therefore, the source attribute uses the following encoding:

**Table 6-32.17 Source Attribute Encoding**

| | |
|---|---|
| $24_{hex}$ | x |

Where x is the Instance ID of the S-Analog Sensor object whose *Value* attribute is the source.

##### 6-32.7.3.1.3 Destination (attribute 12)

Abbreviated Packed EPATH — the *Destination* attribute in this device type is abbreviated as follows:

Logical Segment, Instance Only, 8-bit Logical Address (see Appendix C).

The following defaults apply:

Class ID = 09hex [Discrete Output Point].
Attribute ID = 03 [Value].

Therefore, the Destination attribute uses the following encoding:

**Table 6-32.18 Destination Attribute Encoding**

| | |
|---|---|
| $24_{hex}$ | x |

Where x is the Instance ID of the Discrete Output Point object whose *Value* attribute is the Destination.

### 6-32.8 Defining Device Configuration

Public access to the S-Device Supervisor, S-Gas Calibration, and S-Analog Sensor Objects by the Message Router must be supported for configuration of this device type. There is no Parameter Object defined for access to the device type’s configuration parameters.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-33 Programmable Logic Controller Device

**Device Type: 0E hex**

The Programmable Logic Controller Device type defines a device that acts as a programmable logic controller. No control functions are included in this profile.

## 6-33.1 Object Model

The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-33.1 Objects Present in a Programmable Logic Controller Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |

The Programmable Logic Controller Device profile cannot specify the definition of the Assembly Object or the type of application objects necessary for device operation. This portion of the device profile must be supplied by the product developer as described in Sections 6-1 through 6-7 inclusive.

Since the PLC device profile does not define any static Assembly object instances, use of Assembly object instances in the “Open - Defined by the device profile” ranges (see Table 5A-5.2 Assembly Instance ID Ranges) for vendor specific purposes is permitted.

## 6-33.2 Defining Object Interfaces

The objects in the Programmable Logic Controller Device have the interfaces listed in the following table:

**Table 6-33.2 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-34 ControlNet Physical Layer Component Device

## Device Type: 32 hex

The ControlNet Physical Layer Component shall not be addressable from the link. This device type shall include repeaters and various media for the ControlNet physical layer.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-35 Process Control Valve Device

## Device Type: 1D hex

A Process Control Valve (PCV) is a device that controls a process condition. This can be physically located before or after a process vessel, container or chamber, therefore controlling the entry or exit of fluid / gas to / from a process chamber. It contains a controller that employs closed loop control by receiving a set point and driving the internal actuator such that the process variable is controlled to the set point. The value of the process variable can be obtained either through digital or analog connections. The controller operates in either of two modes. One mode of operation is closed loop process control. The second drives the control valve directly to a position.

## 6-35.1 Object Model

The Object Model in Figure 6-35.1 represents a Process Control Valve Device. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-35.1 Objects Present in a Process Control Valve Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Required | at least 2 Input and 2 Output |
| S-Device Supervisor | Required | 1 |
| S-Analog Sensor | Optional | * |
| S-Analog Actuator | Required | 1 |
| S-Single Stage Controller | Required | 1 to 2 |
| Selection | Conditional ** | 2 |
| Trip Point | Optional | 1 or more |
| Discrete Input Point | Optional | 2 |
| Discrete Output Point | Optional | 1 or more |

* depends on implementation
** Required if two S-Single State Controller Objects are supported.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-35.1 Object Model for the Process Control Valve Device**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-35.2 How Objects Affect Behavior

**Table 6-35.2 Object Affect on Behavior**

| Object | Effect on behavior |
|---|---|
| Identity | Supports the Reset service. Upon receipt of a *Reset* Service Request of any *Type*, the Identity Object sends a *Reset* Service Request to the S-Device Supervisor. |
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines input/output and configuration data format |
| S-Device Supervisor | Supports the Stop, Start, Reset, Abort, Recover and Perform_Diagnostic services for ALL Application Objects in the device and consolidates the Exception Conditions and Application Objects' Status.<br>This object behaves differently from the Identity Object in that the S-Device Supervisor object provides a single point of access to the Application Objects only; it does not affect the DeviceNet specific objects (i.e., Identity, DeviceNet, Connection, etc.). |
| S-Analog Sensor | Monitors the optional analog process variable input(s), and feeds the process variable value to the Single Stage Controller object. Also, optionally monitors the position of the control valve, as well as other signal levels in the device for diagnostic purposes. |
| S-Analog Actuator | Drives the physical valve. |
| S-Single Stage Controller | Calculates the Control Variable required to drive the control valve of the device. Both instances may be present but only one instance is active at any time. |
| Selection | Provides a means of configuring data flow with the device profile. |
| Trip Point | Provides a process trip point comparator for the S-Single Stage Controller Process variable value. Each instance is linked to an S-Single Stage Controller instance. The output of this object may be used to drive the Discrete Output Point object. |
| Discrete Input Point | Monitors the state of the optional limit switches of the device control valve. |
| Discrete Output Point | Reflect status of Trip point object instances. One Discrete Output Point can be assigned to a Trip Point obect. |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-35.3 Defining Object Interfaces

**Table 6-35.3 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| S-Device Supervisor | Assembly or Message Router |
| S-Analog Sensor | Assembly or Message Router |
| S-Analog Actuator | Assembly or Message Router |
| S-Single Stage Controller | Assembly or Message Router |
| Selection | Assembly or Message Router |
| Trip Point | Assembly or Message Router |
| Discrete Input Point | Assembly or Message Router |
| Discrete Output Point | Message Router |

### 6-35.3.1 I/O Assembly Instances

The following table identifies the I/O assembly instances supported by the PCV.

The S-Analog Sensor and S-Single Stage Controller object definition specifies a behavior that modifies the Data Type of certain attributes based upon the first valid I/O connection established to an Assembly Object instance. In order to maintain consistency, this device type will only allow connections to either INT or REAL based Assembly instances. Once a valid connection is established, attempts to configure connections, or otherwise access data, to a different type of Assembly instance will return a RESOURCE UNAVAILABLE error.

**Table 6-35.4 I/O Asembly Instances**

| Number | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | N | Input | 0-1 | INT Process Variable | | | | | | | |
| 2 | Y<br>(default) | Input | 0<br>1-2 | Exception Status<br>Process Variable | | | | | | | |
| 3 | N | Input | 0<br>1-2<br>3-4 | Exception Status<br>Process Variable<br>Valve | | | | | | | |
| 4 | N | Input | 0<br>1-2<br>3-4 | Exception Status<br>Process Variable<br>Control Setpoint | | | | | | | |
| 5 | N | Input | 0<br>1-2<br>3-4<br>5-6 | Exception Status<br>Process Variable<br>Control Setpoint<br>Valve | | | | | | | |
| 6 | N | Input | 0<br>1-2<br>3-4<br>5<br>6-7 | Exception Status<br>Process Variable<br>Control Setpoint<br>Control Mode<br>Valve | | | | | | | |
| 7 | Y<br>(default) | Output | 0-1<br>2-3 | Control Setpoint<br>Control Instance | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Process Control Valve Device, Type: $1D_{Hex}$

| Number | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | N | Output | 0<br>1-2<br>3-4 | Control Mode<br>Control Setpoint<br>Control Instance | | | | | | | |
| 9 | N | Output | 0<br>1-2<br>3-4<br>5-6 | Control Mode<br>Control Setpoint<br>Control Instance<br>Process Variable | | | | | | | |
| 10 | N | Input | 0 | Exception Status | | | | | | | |
| 11 | N | Input | 0<br>1-2<br>3-4 | Exception Status<br>Process Variable<br>Valve | | | | | | | |
| | | | 5 | | | | | | | Dicsrete Input 2 | Discrete Input 1 |
| 12 | N | Input | 0<br>1-2<br>3-4 | Exception Status<br>Process Variable<br>Flow | | | | | | | |
| 13 | N | Input | 0<br>1-2<br>3-4<br>5-6 | Exception Status<br>Process Variable<br>Valve<br>Flow | | | | | | | |
| 14 | N | Output | 1-2 | Process Variable | | | | | | | |
| 15 | N | Input | 0<br>1-2<br>3-4 | Exception Status<br>Process Variable<br>Temperature | | | | | | | |
| 16 | N | Input | 0<br>1-2<br>3-4<br>5-6 | Exception Status<br>Process Variable<br>Flow<br>Temperature | | | | | | | |
| 17 | N | Input | 0-3 | FP-Process Variable | | | | | | | |
| 18 | Y | Input | 0<br>1-4 | Exception Status<br>FP-Process Variable | | | | | | | |
| 19 | N | Input | 0<br>1-4<br>5-8 | Exception Status<br>FP-Process Variable<br>FP- Valve | | | | | | | |
| 20 | N | Input | 0<br>1-4<br>5-8 | Exception Status<br>FP- Process Variable<br>FP- Control Setpoint | | | | | | | |
| 21 | N | Input | 0<br>1-4<br>5-8<br>9-12 | Exception Status<br>FP-Process Variable<br>FP-Control Setpoint<br>FP- Valve | | | | | | | |
| 22 | N | Input | 0<br>1-4<br>5-8<br>9<br>10-13 | Exception Status<br>FP-Process Variable<br>FP-Control Setpoint<br>Control Mode<br>FP- Valve | | | | | | | |
| 23 | Y | Output | 0-3<br>4-5 | FP-Control Setpoint<br>Control Instance | | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Process Control Valve Device, Type: $1D_{Hex}$**

| Number | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 24 | N | Output | 0 | Control Mode | | | | | | | |
| | | | 1-4 | FP-Control Setpoint | | | | | | | |
| | | | 5-6 | Control Instance | | | | | | | |
| 25 | N | Output | 0 | Control Mode | | | | | | | |
| | | | 1-4 | FP-Control Setpoint | | | | | | | |
| | | | 5-6 | Control Instance | | | | | | | |
| | | | 7-10 | FP-Process Variable | | | | | | | |
| 26 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1-4 | FP-Process Variable | | | | | | | |
| | | | 5-8 | FP-Valve | | | | | | | |
| | | | 9 | | | | | | | Discrete Input 2 | Discrete Input 1 |
| 27 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1-4 | FP-Process Variable | | | | | | | |
| | | | 5-8 | FP-Flow | | | | | | | |
| 28 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1-4 | FP-Process Variable | | | | | | | |
| | | | 5-8 | FP-Valve | | | | | | | |
| | | | 9-12 | FP-Flow | | | | | | | |
| 29 | N | Output | 0-3 | FP-Process Variable | | | | | | | |
| 30 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1-4 | FP-Process Variable | | | | | | | |
| | | | 5-9 | FP-Temperature | | | | | | | |
| 31 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1-4 | FP-Process Variable | | | | | | | |
| | | | 5-8 | FP-Flow | | | | | | | |
| | | | 9-12 | FP-Temperature | | | | | | | |
| 32 | N | Output | 0-1 | Control Mode | | | | | | | |
| | | | 2-5 | FP-Control Setpoint | | | | | | | |
| | | | 4-9 | Kp | | | | | | | |
| | | | 8-13 | Ki | | | | | | | |
| | | | 12-17 | Kd | | | | | | | |

The manufacturer of a Process Control Valve Device must specify which optional Assembly instances are supported by the device.

### 6-35.3.2 Mapping I/O Assembly Data Attribute Components

The Data Type of the objects in this profile are based upon the first valid I/O Connection established to an Assembly Object instance. See text of 6-35.4.1.

The following table indicates the I/O assembly Data attribute mapping for this PCV device.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-35.5 I/O Assembly Data Mapping**

| Data Component | Class | | Instance | Attribute | | |
|---|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number | Type |
| Process Variable | S-Single Stage Controller | $33_{hex}$ | 1 | Process Variable | 7 | * |
| Valve | S-Analog Sensor | $31_{hex}$ | 3 | Value | 6 | ** |
| Flow | S-Analog Sensor | $31_{hex}$ | 4 | Value | 6 | ** |
| Temperature | S-Analog Sensor | $31_{hex}$ | 5 | Value | 6 | ** |
| Control Mode | S-Single Stage Controller | $33_{hex}$ | 1 | Control Mode | 5 | USINT |
| Control Setpoint | Selection | 2Ehex | 1 | input_data_value | 15 | * |
| Control Instance | Selection | $2E_{hex}$ | 1 | destination_used | 14 | UINT |
| Status | S-Device Supervisor | $30_{hex}$ | 1 | Exception Status | 12 | BYTE |
| Discrete Input *N* | Discrete Input Point | $08_{hex}$ | *N* | Value | 3 | BOOL |
| Kd | Derivative Gain | $33_{hex}$ | 1 | Kd | 92 | REAL |
| Ki | Integral Gain | $33_{hex}$ | 1 | Ki | 93 | REAL |
| Kp | Poportional Gain | $33_{hex}$ | 1 | Kp | 94 | REAL |

* As specified by S-Single Stage Controller *Data Type* attribute.

** As specified by S-Analog Sensor *Data Type* attribute.

## 6-35.4 Object Limitations and Specific Definitions

### 6-35.4.1 S-Device Supervisor Object

The following attribute limitations apply:

**Table 6-35.6 Device Type Attribute Limitations**

| Attribute | Limitation |
|---|---|
| Device Type | Supported Values:<br>"PCV" = Process Variable Control Valve device |

#### 6-35.4.1.1 Exception Attributes Definition

The following table specifies the data attribute bit mapping for the Device Exception Detail bytes for this PCV device. For more descriptive information, refer to the definition of the S-Device Supervisor Object Class (and, in particular, the Semantics of the Exception Detail Alarm and Exception Detail Warning attributes). Noted, for each entry, is the Object and Instance from which the Status byte/bit is mapped. Refer to the object specification for the detailed bit mapping.

Any Exception Bit not supported must default to 0. Note that this profile allows for only one byte of manufacturer specific exception detail.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-35.7 Data Attribute Bit Mapping**

| Data Component Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PCV Device Exception Detail Size | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| PCV Device Exception Detail Byte #0 | Process Variable High S-Analog Sensor (2) | Process Variable Low S-Analog Sensor (2) | Not Reading Valid * S-Analog Sensor (2) | Position Control S-Single Stage Controller (2) | Process Variable Control S-Single Stage Controller (1) | Process Variable High S-Analog Sensor (1) | Process Variable Low S-Analog Sensor (1) | Not Reading Valid * S-Analog Sensor (1) |
| PCV Device Exception Detail Byte #1 | Temp. Low S-Analog Sensor (5) | Not Reading Valid * S-Analog Sensor (5) | Flow High S-Analog Sensor (4) | Flow Low S-Analog Sensor (4) | Not Reading Valid * S-Analog Sensor (4) | Valve High S-Analog Sensor (3) | Valve Low S-Analog Sensor (3) | Not Reading Valid * S-Analog Sensor (3) |
| PCV Device Exception Detail Byte #2 | Not Reading Valid * S-Analog Sensor (8) | Auxiliary Input High S-Analog Sensor (7) | Auxiliary Input Low S-Analog Sensor (7) | Not Reading Valid * S-Analog Sensor (7) | DeviceNet 24V High ** S-Analog Sensor (6) | DeviceNet 24V Low ** S-Analog Sensor (6) | Not Reading Valid * S-Analog Sensor (6) | Temp. High S-Analog Sensor (5) |
| PCV Device Exception Detail Byte #3 | Device -15V High S-Analog Sensor (10) | Device -15V Low S-Analog Sensor (10) | Not Reading Valid * S-Analog Sensor (10) | Device +15V High S-Analog Sensor (9) | Device +15V Low S-Analog Sensor (9) | Not Reading Valid * S-Analog Sensor (9) | Logic 5V High S-Analog Sensor (8) | Logic 5V Low S-Analog Sensor (8) |
| PCV Device Exception Detail Byte #4 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Status Discrete Input Point (2) | Status Discrete Input Point (1) |
| Manufacturer Exception Detail Size | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Manufacturer Exception Detail | 8 Bits defined by Manufact-urer | | | | | | | |

* Only used in the Warning Exception Detail, this bit is always = 0 in the Alarm Exception Detail. 'Not' indicates the use of the negation of 'Reading Valid'.

** Not supported if CIP network is not DeviceNet. Shall be zero.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-35.4.2 S-Analog Sensor Object

This section provides the additional components and specifications for this object as it applies to this profile.

The temperature instance is location specific to each manufacturer, if supported. The physical location of the actual temperature sensor the instance reports is determined by the manufacturer.

The following instance limitations apply:

**Table 6-35.8 Instance Limitations**

| Instance | Description | Subclass |
|---|---|---|
| 1 | Process Variable Input #1 (Low Range) | 06 * |
| 2 | Process Variable Input #2 (High Range) | 06 * |
| 3 | Control Valve Position Feedback | none |
| 4 | Flow Sensor | none |
| 5 | Temperature Sensor | none |
| 6 | DeviceNet 24V Power Input | none |
| 7 | Auxiliary Voltage Input | none |
| 8 | Logic 5V | none |
| 9 | Device +15V | none |
| 10 | Device -15V | none |
| * S-Analog Sensor Object Instance Subclass 06 - Transfer Function | | |

The following attribute limitations apply:

**Table 6-35.9 Attribute Limitations**

| Attribute | Instance | Limitation | Requirements |
|---|---|---|---|
| Data Type | All | Supported Values = {INT; REAL}<br>Access Rule = Get only, after an assembly instance connection is established | INT |
| Data Units | 1,2<br>3<br>4<br>5<br>6,7,8,9,10 | Supported Values = {Counts, Percent, Torr, mTorr, Bar, mBar, Pa, kPa}<br>Supported Values = {Counts, Percent, Degrees}<br>Supported Values = {Counts, Percent, SCCM, SLM}<br>Supported Values = {Counts, Centigrade, Fahrenheit}<br>Supported Values = {Counts, Percent, Volts} | Counts |
| Offset-A Data Type | All | Supported Values = {INT; REAL} | INT |
| Gain Data Type | All | Supported Values = {INT; REAL} | INT |
| Value | All | For:<br>Data Units = Counts<br>Data Type = INT<br>7FFF = 140% of Full Scale Sensor Reading<br>0000 = 0<br>8000 = -140% of Full Scale Sensor Reading | |

### 6-35.4.3 S-Single Stage Controller Object

The following instance limitations apply:



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-35.10 Instance Limitations**

| Instance | Description | Subclass |
|---|---|---|
| 1 | Process Variable Control | 01 * |
| 2 | Control Valve Position | none |
| * S-Single Stage Controller Object Instance Subclass 01 - PID and Source Select | | |

The following attribute limitations apply:

**Table 6-35.11 Attribute Limitations**

| Attribute | Limitation | Requirements |
|---|---|---|
| Data Type | Supported Values = {INT; REAL}<br>Access Rule = Get only, after an assembly instance connection is established | INT |
| Data Units / Instance 1 | Supported Values = {Counts, Percent, Torr, mTorr, Bar, mBar, Pa, kPa} | Counts |
| Data Units / Instance 2 | Supported Values = {Counts, Percent, Degrees}<br>0 = closed | Counts |
| Process Variable | | Shall be supported |

### 6-35.4.4 Selection Object

The following instance limitations apply:

Where supported, this profile requires the Selection Object Revision 2 or higher.

**Table 6-35.12 Instance Limitations**

| Instance | Description |
|---|---|
| 1 | Setpoint |
| 2 | Control Variable |

The following attribute values shall be assigned for Instance 1:

**Table 6-35.13 Attribute Limitations – Instance 1**

| Attribute | Limitation (assigned values are shown in hex format) |
|---|---|
| max_destinations | 02 |
| number_of_destinations | 02 |
| destination_list | 06 / 20 / 33 / 24 / 01 / 30 / 06 — S-Single Stage Controller, Instance 1, setpoint<br>06 / 20 / 33 / 24 / 02 / 30 / 06 — S-Single Stage Controller, Instance 2, setpoint<br>Access Rule = Get Only |
| max_sources | 00 |
| number_of_sources | 00 |
| source_list | Not Supported |
| source_used | 00 |
| algorithm_type | 04 — Programmable Data Flow<br>Access Rule = Get Only |
| object_source_list | 00<br>Access Rule = Get Only |
| destination_used | Range = 0 - 2 |

The following attribute values shall be assigned for Instance 2:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This object instance is not able to be configured over the network. Its purpose is to function as an automatic Data Flow selection mechanism.

**Table 6-35.14 Attribute Limitations – Instance 2**

| Attribute | Limitation (assigned values are shown in hex format) |
|---|---|
| max_destinations | 01 |
| number_of_destinations | 01 |
| destination_list | 06 / 20 / 32 / 24 / 01 / 30 / 06 — S-Analog Actuator, Instance 1, Value<br>Access Rule = Get Only |
| max_sources | 02 |
| number_of_sources | 02 |
| source_list | Not Supported |
| source_used | Range = 1 - 2<br>Access Rule = Get Only |
| algorithm_type | 04 — Programmable Data Flow<br>Access Rule = Get Only |
| object_source_list | 06 / 20 / 33 / 24 / 01 / 30 / 09 — S-Single Stage Controller, Instance 1, Control Variable<br>06 / 20 / 33 / 24 / 02 / 30 / 09 — S-Single Stage Controller, Instance 2, Control Variable<br>Access Rule = Get Only |
| destination_used | 01<br>Access Rule = Get Only |

### 6-35.4.5 Discrete Input Point Object

The following instances shall be applied as follows:

**Table 6-35.15 Instances**

| Instance | Description |
|---|---|
| 1 | Close Limit Switch |
| 2 | Open Limit Switch |

It is recommended that these object instances support the Off_On Cycles attribute.

## 6-35.5 Defining Device Configuration

Public access to the S-Device Supervisor, S-Analog Sensor, and S-Single Stage Controller Objects by the Message Router must be supported for configuration of this device type.

For the *Data Units* for the *S-Analog Sensor Object* instances one through four are for process inputs, the *S-Single Stage Controller Object*'s *Setpoint* and the *S-Single Stage Controller Object*'s *Process Variable* will be self-modifying for any change in *Data Units* to any other of these attributes. The self-modified value shall give consistency of *Data Units* to all these instances. This is defined by the class level attributes of Control Insurance and Control Setpoint.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-36 Turbomolecular Vacuum Pump Device

**Device Type: 21 hex**

A Turbomolecular Vacuum Pump (Turbo Pump) is a type of vacuum pump. A vacuum pump is a device that is used for pumping air and other gasses for the purposes of generating negative pressures on its inlet port.

The Turbo Pump device is composed of a motor, a mechanical (turbomolecular) pumping apparatus and various electronic control elements. Typically, thermal management represents another component of the device.

## 6-36.1 Object Model

The Object Model in Figure 6-36.1 represents a Turbo Pump Device. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-36.1 Objects Present in a Turbo Pump Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Required | at least 2 Input and 2 Output |
| S-Device Supervisor | Required | 1 |
| Register | Optional | 0 or 1 |
| Discrete Input Point | Required | 1 |
| Discrete Output Point | Required | 1, 2 or 3 |
| AC/DC Drive | Required | 1 |
| S-Analog Sensor | Optional | as many as 5 |
| S-Single Stage Controller | Optional | 0 or 1 |

## 6-36.2 Class Subclasses

Each class level subclass defines a unique meaning for an overlapping range of class attribute IDs and/or class service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given class is identified by the value of its Subclass class attribute.

There are no Class Level Subclasses specified for the Turbo Pump device.

## 6-36.3 Instance Subclasses

Each instance level subclass defines a unique meaning for an overlapping range of instance attribute IDs and/or instance service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given instance is identified by the value of its Subclass instance attribute.

There are no Instance Level Subclasses specified for the Turbo Pump device.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-36.1 Object Model for the Turbo Pump Device**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-36.4 How Objects Affect Behavior

**Table 6-36.2 Object Affect on Behavior**

| Object | Effect on behavior |
|---|---|
| Identity | Supports the Reset service. Upon receipt of a Reset Service request, a similar request is passed to the S-Device Supervisor. |
| CIP Required | See Section 6-2.2 for details. |
| Assembly | Defines input/output and configuration data format |
| S-Device Supervisor | Supports the Stop, Start, Reset, Abort, Recover and Perform_Diagnostic services for ALL Application Objects in the device and consolidates the Exception Conditions and Application Objects' Status |
| Register | Vent Valve configuration settings |
| Discrete Input Point | No effect |
| Discrete Output Point | Turns pump on and off and affects the states of the Bakeout Heater and Purge Valve. The value of Discrete Output Point Object instance 1, Value attribute (ID 3) is sent to AC/DC Drive Object instance 1, Speed Control attribute (ID 38) Bit 0 (Run Request). |
| AC/DC Drive | Manages the operation of the pump motor. The value of the Speed Status attribute (ID 39) Bit 0 (Running) is sent to Discrete Input Point object instance 1, Value attribute (ID 3). |
| S-Analog Sensor | Alarm Trip Points may cause the pump to shut down as specified. Feeds values to the Process Variables of S-Single Stage Controllers. |
| S-Single Stage Controller | Control various temperature zones in the pump assembly |

## 6-36.5 Defining Object Interfaces

**Table 6-36.3 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| S-Device Supervisor | Assembly or Message Router |
| Register | Assembly or Message Router |
| Discrete Input Point | Assembly or Message Router |
| Discrete Output Point | Assembly or Message Router |
| AC/DC Drive | Assembly or Message Router |
| S-Analog Sensor | Assembly or Message Router |
| S-Single Stage Controller | Assembly or Message Router |

## 6-36.6 I/O Assembly Instances

The manufacturer of a Turbo Pump Device must specify which Assembly instances are supported by the device.

The following table identifies the I/O assembly instances.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-36.4 I/O Assembly Instances**

| Instance | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Y (default) | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Speed Status | | | | | | | |
| | | | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Pump On Status |
| 2 | Y | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Speed Status | | | | | | | |
| | | | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Pump On Status |
| | | | 3 - 4 | Pump Speed | | | | | | | |
| 3 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Speed Status | | | | | | | |
| | | | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Pump On Status |
| | | | 3 - 4 | Pump Speed | | | | | | | |
| | | | 5 - 6 | Coil Temp | | | | | | | |
| | | | 7 - 8 | Current | | | | | | | |
| 4 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Speed Status | | | | | | | |
| | | | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Pump On Status |
| | | | 3 - 4 | Pump Speed | | | | | | | |
| | | | 5 - 6 | Inlet Pressure | | | | | | | |
| 5 | Y (default) | Output | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Pump On |
| 6 | Y | Output | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Pump On |
| | | | 1 | Speed Control | | | | | | | |
| 7 | N | Output | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Pump On |
| | | | 1 | Speed Control | | | | | | | |
| | | | 2 - 3 | Speed Target | | | | | | | |
| 8 | N | Output | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Pump On |
| | | | 1 | Speed Control | | | | | | | |
| | | | 2 - 3 | Pressure Target | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Instance | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | N | Output | 0 | Speed Scale | | | | | | | |
| | | | 1 - 2 | Speed Data Units | | | | | | | |
| | | | 3 - 4 | High Speed Limit | | | | | | | |
| | | | 5 - 6 | Speed Target Data Units | | | | | | | |
| | | | 7 - 8 | Speed Target | | | | | | | |
| | | | 9 | Time Scale | | | | | | | |
| | | | 10 - 11 | Accelerate Time | | | | | | | |
| | | | 12 - 13 | Deceleration Time | | | | | | | |
| | | | 14 - 15 | Speed Trip Time | | | | | | | |
| | | | 16 | Current Scale | | | | | | | |
| | | | 17 - 18 | Current Limit | | | | | | | |
| 10 | N | Output | 0 | Drive Mode | | | | | | | |
| | | | 1 | Process Scale | | | | | | | |
| | | | 2 - 3 | Process Data Units | | | | | | | |
| | | | 4 - 5 | Pressure Target | | | | | | | |
| 11 | N | Output | 0 | Coil Temp Control Mode | | | | | | | |
| | | | 1 - 2 | Coil Temp Setpoint | | | | | | | |
| | | | 3 | Case Temp Control Mode | | | | | | | |
| | | | 4 - 5 | Case Temp Setpoint | | | | | | | |
| | | | 6 | Bearing Temp Control Mode | | | | | | | |
| | | | 7 - 8 | Bearing Temp Setpoint | | | | | | | |
| 12 | N | Output | 0 - 1 | Vent Valve Config | | | | | | | |

All of the elemental components listed above follow standard CIP Data Encoding as specified in Appendix C.

Any component that is unsupported shall be set to 0.

## 6-36.7 I/O Assembly Instance Component Mapping

The following table indicates the I/O assembly Data attribute mapping for this device.

**Table 6-36.5 I/O Assembly Data Mapping**

| Data Component | Object Name | Class ID | Instance ID | Attribute Name | Attrib | Data Type |
|---|---|---|---|---|---|---|
| Exception Status | S-Device Supervisor | $30_{hex}$ | 1 | Exception Status | 12 | BYTE |
| Pump On Status | Discrete Input Point | $08_{hex}$ | 1 | Value | 3 | BOOL |
| Pump On | Discrete Output Point | $09_{hex}$ | 1 | Value | 3 | BOOL |
| Pump Speed | AC/DC Drive | $2A_{hex}$ | 1 | SpeedActual | 7 | INT |
| Speed Status | AC/DC Drive | $2A_{hex}$ | 1 | Speed Status | 42 | BYTE |
| Speed Control | AC/DC Drive | $2A_{hex}$ | 1 | Speed Control | 41 | BYTE |
| Speed Target | AC/DC Drive | $2A_{hex}$ | 1 | SpeedRef | 8 | INT |
| Current | AC/DC Drive | $2A_{hex}$ | 1 | CurrentActual | 9 | INT |
| Inlet Pressure | AC/DC Drive | $2A_{hex}$ | 1 | ProcessActual | 13 | INT |
| Speed Scale | AC/DC Drive | $2A_{hex}$ | 1 | SpeedScale | 22 | SINT |
| Speed Data Units | AC/DC Drive | $2A_{hex}$ | 1 | SpeedActual Data Units | 43 | ENGUNITS |
| High Speed Limit | AC/DC Drive | $2A_{hex}$ | 1 | HighSpeed-Limit | 21 | UINT |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Turbomolecular Vacuum Pump Device, Type: $21_{Hex}$**

| Data Component | Object Name | Class ID | Instance ID | Attribute Name | Attrib | Data Type |
|---|---|---|---|---|---|---|
| Speed Target Data Units | AC/DC Drive | $2A_{hex}$ | 1 | SpeedRef Data Units | 44 | ENGUNITS |
| Time Scale | AC/DC Drive | $2A_{hex}$ | 1 | TimeScale | 28 | SINT |
| Acceleration Time | AC/DC Drive | $2A_{hex}$ | 1 | AccelTime | 18 | UINT |
| Deceleration Time | AC/DC Drive | $2A_{hex}$ | 1 | DecelTime | 19 | UINT |
| Overload Time | AC/DC Drive | $2A_{hex}$ | 1 | Speed Trip Time | 38 | UINT |
| Current Scale | AC/DC Drive | $2A_{hex}$ | 1 | CurrnetScale | 23 | SINT |
| Current Limit | AC/DC Drive | $2A_{hex}$ | 1 | CurrentLimit | 10 | INT |
| Drive Mode | AC/DC Drive | $2A_{hex}$ | 1 | DriveMode | 6 | USINT |
| Process Scale | AC/DC Drive | $2A_{hex}$ | 1 | ProcessScale | 25 | SINT |
| Process Data Units | AC/DC Drive | $2A_{hex}$ | 1 | Process Data Units | 45 | ENGUNITS |
| Pressure Target | AC/DC Drive | $2A_{hex}$ | 1 | ProcessRef | 14 | INT |
| Coil Temp | S-Analog Sensor | $31_{hex}$ | 1 | Value | 6 | INT |
| Coil Temp Control Mode | S-Single Stage Controller | $33_{hex}$ | 1 | Control Mode | 5 | USINT |
| Coil Temp Setpoint | S-Single Stage Controller | $33_{hex}$ | 1 | Setpoint | 6 | INT |
| Case Temp Control Mode | S-Single Stage Controller | $33_{hex}$ | 2 | Control Mode | 5 | USINT |
| Case Temp Setpoint | S-Single Stage Controller | $33_{hex}$ | 2 | Setpoint | 6 | INT |
| Bearing Temp Control Mode | S-Single Stage Controller | $33_{hex}$ | 3 | Control Mode | 5 | USINT |
| Bearing Temp Setpoint | S-Single Stage Controller | $33_{hex}$ | 3 | Setpoint | 6 | INT |
| Vent Valve Config | Register | $07_{hex}$ | 1 | Data | 4 | ARRAY of BOOL |

## 6-36.8 Object Limitations and Specific Definitions

This section describes limitations and specific definitions applicable to the listed objects of this device profile when these objects are applied in this type of device.

### 6-36.8.1 S-Device Supervisor

The following attribute limitations apply:

**Table 6-36.6 Device Type Attribute Limitations**

| Attribute | Requirements |
|---|---|
| Device Type | Set to "TVP" for the Turbomolecular Vacuum Pump Device |

The following table specifies the data attribute bit mapping for the **Exception Detail** bytes for this device. For more descriptive information, see the definition of the S-Device Supervisor Object Class.

Any Exception Bit not supported shall default to 0.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-36.7 Attribute Bit Mapping**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Turbo Pump Device Exception Detail Size | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Turbo Pump Device Exception Detail Byte 0 | Excess Vibration | Startup Timeout * | Speed Trip | Over-current * | Over-speed * | Mains Failure * | TMS Failure | CNT Failure |
| Turbo Pump Device Exception Detail Byte 1 | 0 | Inter-locked | Bearing Fault | Cable Fault * | Controller Overheat | Bearing Overheat | Case Overheat | Motor Coil Overheat |
| Manufacturer Exception Detail Size ** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 ** |
| Manufacturer Exception Detail ** | 8 Bits defined by Manufacturer | | | | | | | |

* Exception Detail Alarm only; not supported in Exception Detail Warning

** This example shows a one-byte manufacturer exception detail, but there may be zero or many bytes defined

**Table 6-36.8 Exception Detail Bit Description**

| Exception | Description |
|---|---|
| CNT Failure | Controller Unit failure |
| TMS Failure | Temperature Management System failure |
| Mains Failure | Insufficient power (power failure or incorrect connection) |
| Over-speed | Pump failure |
| Over-current | Pump failure |
| Speed Trip | Pump overload typically sensed by reduced speed exceeding the Speed Trip Time |
| Startup Timeout | Normal speed from startup not obtained within specified AccelTime |
| Excess Vibration | Physical fault |
| Motor Coil Overheat | Thermal fault |
| Case Overheat | Thermal fault |
| Bearing Overheat | Thermal fault |
| Controller Overheat | Thermal fault |
| Cable Fault | Controller Unit to Pump interconnection cable is misconnected or disconnected |
| Bearing Fault | Damaged bearing requires service |
| Interlocked | Normal pump operation is not possible due to an external interlock signal |

## 6-36.8.2 Register Object

### 6-36.8.2.1 Object Instance Number Assignment

The following instance numbering shall apply:



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-36.9 Instance Numbers**

| Instance | Required | Type | Description |
|---|---|---|---|
| 1 | No | Output | Vent Valve Configuration |

**Table 6-36.10 Instance 1 — Output — Vent Valve Configuration Data**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Valve 4 | | Valve 3 | | Valve 2 | | Valve 1 | |
| 1 | Valve 8 | | Valve 7 | | Valve 6 | | Valve 5 | |

The following values are defined for each *Valve* listed above.

**Table 6-36.11 Values Defined For Each Valve**

| Valve n Config Nibble Value | Description |
|---|---|
| 0 | Disabled / No Vent |
| 1 | Pump Controlled Vent |
| 2 | Vendor Specified Vent |
| 3 | Reserved by DeviceNet |

### 6-36.8.3 Discrete Input Point Object

#### 6-36.8.3.1 Object Instance Number Assignment

The following instance numbering shall apply:

**Table 6-36.12 Instance Numbering**

| Instance | Required | Name | Description |
|---|---|---|---|
| 1 | Yes | Pump On/Off | 0 = Pump Off<br>1 = Pump On AND Speed > 0 |

#### 6-36.8.3.2 Attribute Definitions

The following attribute limitations apply:

**Table 6-36.13 Attribute Limitations**

| Attribute | Limitation |
|---|---|
| Off_On Cycles | Supported with Get Only Access Rule |

The value of Discrete Input Point Object instance 1, Value attribute (ID 3) is received from AC/DC Drive Object instance ID 1, Speed Status attribute (ID 39) Bit 0 (Running).

### 6-36.8.4 Discrete Output Point Object

#### 6-36.8.4.1 Object Instance Number Assignment

The following instance numbering shall apply:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-36.14 Instance Numbering**

| Instance | Required | Name | Description of the Value Attribute |
|---|---|---|---|
| 1 | Yes | Pump On/Off | 0 = Pump Off<br>1 = Pump On |
| 2 | No | TMS Function or<br>Bakeout Heater * | 0 = TMS or Heater Off<br>1 = TMS or Heater On |
| 3 | No | Purge Gas Valve | 0 = Valve Closed<br>1 = Valve Open |
| * TMS or Bakeout Heater is based on Pump Configuration | | | |

The value of Discrete Output Point Object instance 1, Value attribute (ID 3) is sent to AC/DC Drive Object instance ID 1, Speed Control attribute (ID 38) Bit 0 (Run Request).

## 6-36.8.5 AC/DC Drive Object

### 6-36.8.5.1 Attribute Limitations

The following table lists the attributes of the AC/DC Drive Object, Instance 1, and specifies any special definitions for this device profile. Attributes implemented as specified are not shown in this table.

**Table 6-36.15 AC/DC Drive Object, Instance 1 Attributes**

| Attribute | Limitation * | Default | Definition |
|---|---|---|---|
| NetRef | Optional Support<br>Access Rule = Get | 1 | 0 = Speed Control NOT by Net<br>1 = Speed Control by Net |
| NetProc | Optional Support<br>Access Rule = Get | 1 | 0 = Process Control NOT by Net<br>1 = Process Control by Net |
| DriveMode | Supported Values =<br>0 = Vendor Specified<br>2 = Closed Loop Speed<br>4 = Process Control | 2 | |
| SpeedActual | Units Specified by Speed Data Units attribute value | | |
| SpeedRef | Units Specified by Speed Data Units attribute value | | |
| ProcessActual | Units Specified by Process Data Units attribute value | | Value derived from Gauge at Pump Inlet |
| ProcessRef | Units Specified by Process Data Units attribute value | | |
| OutputVoltage | Not Supported | | |
| RefFromNet | Optional Support<br>Access Rule = Get | 1 | 0 = Local Speed Reference<br>1 = Net Speed Reference |
| ProcFromNet | Optional Support<br>Access Rule = Get | 1 | 0 = Local Process Reference<br>1 = Net Process Reference |
| Speed Trip Time | As defined | | a Speed Trip Condition results in the setting of the Speed Trip bit of the S-Device Supervisor Exception Detail. |
| Speed Status | Required | | |
| SpeedActual Data Units | Limited to:<br>Units of Frequency (see Appendix D) as well as the following special vales:<br>1007(Hex) = % of Target<br>0800(Hex) = % of MaxRated | $1F0F_{hex}$ | If attribute not supported the value of "RPM" shall be specified. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Attribute | Limitation * | Default | Definition |
|---|---|---|---|
| SpeedRef Data Units | Limited to:<br>Units of Frequency (see Appendix D) | $1F0F_{hex}$ | If attribute not supported the value of "RPM" shall be specified. |
| Process Data Units | Limited to Units of Pressure (see Appendix D) | $1309_{hex}$ | If attribute not supported the value of "Pa" shall be specified. |
| * Vendor may impose further limitations. | | | |

### 6-36.8.5.2 Behavior

The value of Discrete Input Point Object instance 1, Value attribute (ID 3) is received from AC/DC Drive Object instance ID 1, Speed Status attribute (ID 39) Bit 0 (Running).

The value of Discrete Output Point Object instance 1, Value attribute (ID 3) is sent to AC/DC Drive Object instance ID 1, Speed Control attribute (ID 38) Bit 0 (Run Request).

The behavior associated with the Speed Trip Condition and the Acceleration Timeout shall be that the respective bit is set in the S-Device Supervisor *exception detail* attribute to indicate the condition. Further, based upon the vendor's declaration of these bits as Critical or Non-Critical, the behavior of the Turbo shall be affected as specified by the vendor's documentation. Typically, either of these conditions will be considered Critical and cause the Turbo to shutdown.

## 6-36.8.6 S-Analog Sensor Object

### 6-36.8.6.1 Object Instance Number Assignment

The following instance numbering shall apply:

**Table 6-36.16 Instance Number Assignments**

| Instance | Description |
|---|---|
| 1 | Motor Coil Temperature |
| 2 | Case Temperature |
| 3 | Bearing Temperature |
| 4 | Controller Unit Temperature |
| 5 | TMS Temperature |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 6-36.8.6.2 Attribute Definitions

The following attribute limitations apply to all instances:

**Table 6-36.17 Attribute Limitations**

| Attribute | Limitation |
|---|---|
| Data Type | Supported Value = {INT} |
| Data Units | Supported Value = {decidegrees Centigrade} |

Alarm Trip Points and Warning Trip Points shall be supported and shall be limited to GET only Access if the corresponding components are supported in the *Exception Detail Alarm* and *Exception Detail Warning* attributes defined for the S-Device Supervisor object.

### 6-36.8.7 S-Single Stage Controller Object

#### 6-36.8.7.1 Object Instance Number Assignment

The following instance numbering shall apply:

**Table 6-36.18 Instance Number Assignment**

| Instance | Description |
|---|---|
| 1 | TMS Control |

#### 6-36.8.7.2 Attribute Definitions

The following attribute limitations apply to all instances:

**Table 6-36.19 Attribute Limitations**

| Attribute | Limitation |
|---|---|
| Data Type | Supported Values = {INT} |
| Data Units | Supported Values = {decidegrees Centigrade} |
| Control Mode | Supported Values = {Normal, & Off} * |

* The S-Single Stage Controller object instance supports only the Normal and Off control modes as defined by the object specification. The Normal control mode behaves as defined. However, the Off control mode behaves in a manner that is vendor specific. The Vendor shall specify the behavior of the object in the Off control mode. Examples include: maximum cooling, no cooling, or default cooling.

Alarm Trip Points and Warning Trip Points shall be supported and shall be limited to GET only Access if the corresponding components are supported in the *Exception Detail Alarm* and *Exception Detail Warning* attributes defined for the S-Device Supervisor object.

## 6-36.9 Defining Device Configuration

Public access to the S-Device Supervisor by the Message Router must be supported for configuration of this device type.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-37 Residual Gas Analyzer Device

**Device Type: 1E hex**

A Residual Gas Analyzer (RGA) is a device that measures partial pressures of gasses. The device is composed of a filament, which is controlled through the S-Device Supervisor object, and a partial pressure measurement apparatus, which is controlled by S-Partial Pressure objects.

The S-Partial Pressure objects are instantiated (created), one each, for every Atomic Mass Unit (AMU) to be scanned. Each instance is configured for the measurement of partial pressure or range of partial pressures. Class Level services are supported for requesting scan lists of partial pressures.

## 6-37.1 Object Model

The Object Model in Figure 6-37.1 represents a Residual Gas Analyzer Device. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-37.1 Objects Present in a Residual Gas Analyzer Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Required | at least 1 Input and 1 Output |
| S-Device Supervisor | Required | 1 |
| S-Partial Pressure | Required | at least 1 * |
| S-Analog Sensor | Optional | 1 |
| Discrete Input Point | Optional | 1 |
| Discrete Output Point | Optional | 1 or 2 |

* The number of instances is determined by the total number of AMU values or gas species supported per scan. Generally, this number will be 100 - 800.

## 6-37.2 Class Subclasses

Each class level subclass defines a unique meaning for an overlapping range of class attribute IDs and/or class service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given class is identified by the value of its Subclass class attribute.

There are no Class Level Subclasses specified for the RGA device.

## 6-37.3 Instance Subclasses

Each instance level subclass defines a unique meaning for an overlapping range of instance attribute IDs and/or instance service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given instance is identified by the value of its Subclass instance attribute.

There are no Instance Level Subclasses specified for the RGA device.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-37.1 Object Model for the RGA Device**

## 6-37.4 How Objects Affect Behavior

**Table 6-37.2 Object Affect on Behavior**

| Object | Effect on behavior |
|---|---|
| Identity | Supports the Reset service. Upon receipt of a Reset Service request, a similar request is passed to the S-Device Supervisor. |
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines input/output and configuration data format |
| S-Device Supervisor | Supports the Stop, Start, Reset, Abort, Recover and Perform_Diagnostic services for ALL Application Objects in the device and consolidates the Exception Conditions and Application Objects' Status |
| S-Partial Pressure | Provides the means by which Partial Pressures are selected, measured and reported. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Object | Effect on behavior |
|---|---|
| S-Analog Sensor | If present, the value from this measurement may interlock the device such that it may not be able to turn on for pressures are that are considered by the manufacturer to be too high |
| Discrete Input Point | No effect on the device |
| Discrete Output Point | No effect on the device |

## 6-37.5 Defining Object Interfaces

**Table 6-37.3 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| S-Device Supervisor | Assembly or Message Router |
| S-Partial Pressure | Assembly or Message Router |
| S-Analog Sensor | Assembly or Message Router |
| Discrete Input Point | Assembly or Message Router |
| Discrete Output Point | Assembly or Message Router |

## 6-37.6 I/O Assembly Instances

The manufacturer of an RGA Device must specify which Assembly instances are supported by the device as well as the maximum number of S-Partial Pressure instances. For devices configured with fewer Partial Pressures than the number specified in a given input assembly, the missing Data Components of that assembly will be set to zero (0). Note that for the purpose of bandwidth optimization, I/O connections to such assemblies, where less than the included number of instances are valid, the I/O connection's *Produced_Connection_Size* attribute value can be set accordingly.

Note that the member lists for Partial Pressures and Status attributes are arranged according to Object Instance ID. Even if a given S-Partial Pressure Instance is disabled, it will be included in the I/O Data. Therefore, care must be taken by the user to specify AMU values for each instance in order of desirability, not necessarily AMU value.

The following table identifies the I/O assembly instances.

**Table 6-37.4 I/O Assembly Instances**

| Instance | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Y (default) | Output | 0 | Filaments On/Off | | | | | | | |
| 2 | Y (default) | Input | 0 | Exception Status | | | | | | | |
| | | | 1 - 6 | Time | | | | | | | |
| | | | 7 - 10 | Partial Pressure 1 | | | | | | | |
| 3 | Y | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Status 1 | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Residual Gas Analyzer Device, Type: $1E_{Hex}$

| Instance | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | Y | Input | 0 | Exception Status | | | | | | | |
| | | | 1 - 6 | Time | | | | | | | |
| | | | 7 - 10 | Partial Pressure 1 | | | | | | | |
| | | | 11 - 14 | Partial Pressure 2 | | | | | | | |
| | | | 15 - 18 | Partial Pressure 3 | | | | | | | |
| | | | 19 - 22 | Partial Pressure 4 | | | | | | | |
| | | | 23 - 26 | Partial Pressure 5 | | | | | | | |
| | | | 27 - 30 | Partial Pressure 6 | | | | | | | |
| | | | 31 - 34 | Partial Pressure 7 | | | | | | | |
| | | | 35 - 38 | Partial Pressure 8 | | | | | | | |
| | | | 39 - 42 | Partial Pressure 9 | | | | | | | |
| | | | 43 - 46 | Partial Pressure 10 | | | | | | | |
| 5 | Y | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Status 1 | | | | | | | |
| | | | 2 | Status 2 | | | | | | | |
| | | | 3 | Status 3 | | | | | | | |
| | | | 4 | Status 4 | | | | | | | |
| | | | 5 | Status 5 | | | | | | | |
| | | | 6 | Status 6 | | | | | | | |
| | | | 7 | Status 7 | | | | | | | |
| | | | 8 | Status 8 | | | | | | | |
| | | | 9 | Status 9 | | | | | | | |
| | | | 10 | Status 10 | | | | | | | |
| 6 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 - 6 | Time | | | | | | | |
| | | | 7 - 10 | Partial Pressure 1 | | | | | | | |
| | | | 11 - 14 | Partial Pressure 2 | | | | | | | |
| | | | 15 - 18 | Partial Pressure 3 | | | | | | | |
| | | | - - - | - - - | | | | | | | |
| | | | - - - | - - - | | | | | | | |
| | | | 403-406 | Partial Pressure 100 | | | | | | | |
| 7 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Status 1 | | | | | | | |
| | | | 2 | Status 2 | | | | | | | |
| | | | 3 | Status 3 | | | | | | | |
| | | | - - - | - - - | | | | | | | |
| | | | - - - | - - - | | | | | | | |
| | | | 100 | Status 100 | | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Residual Gas Analyzer Device, Type: $1E_{Hex}$**

| Instance | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 - 6 | Time | | | | | | | |
| | | | 7 - 10 | Partial Pressure 1 | | | | | | | |
| | | | 11 - 14 | Partial Pressure 2 | | | | | | | |
| | | | 15 - 18 | Partial Pressure 3 | | | | | | | |
| | | | - - - | - - - | | | | | | | |
| | | | - - - | - - - | | | | | | | |
| | | | 3203-3206 | Partial Pressure 800 | | | | | | | |
| 9 | N | Input | 0 | Exception Status | | | | | | | |
| | | | 1 | Status 1 | | | | | | | |
| | | | 2 | Status 2 | | | | | | | |
| | | | 3 | Status 3 | | | | | | | |
| | | | - - - | - - - | | | | | | | |
| | | | - - - | - - - | | | | | | | |
| | | | 800 | Status 800 | | | | | | | |

All of the elemental components listed above follow standard CIP Data Encoding as specified in Appendix C.

### 6-37.6.1 I/O Assembly Instance Component Formats

All of the elemental components listed above follow standard CIP Data Encoding as specified in Appendix C. The encoding order for all data is Low Byte first. All members are LSB aligned and end on Byte boundaries. That is, upper bits are stuffed with 0 as needed to fill to the next byte.

### 6-37.6.2 I/O Assembly Instance Component Mapping

The following table indicates the I/O assembly Data attribute mapping for this device.

**Table 6-37.5 I/O Assembly Data Mapping**

| Data Component Name | Object Name | Class ID | Instance ID | Attribute Name | Attribute ID | Member Index | Data Type |
|---|---|---|---|---|---|---|---|
| Filaments On/Off | S-Partial Pressure | 38hex | 0 ** | Filament Control | 32 | N.A. | BYTE |
| Exception Status | S-Device Supervisor | 30hex | 1 | Exception Status | 12 | N.A. | BYTE |
| Time | S-Device Supervisor | 30hex | 1 | Time | 17 | N.A. | DATE_ AND_ TIME |
| Partial Pressure i * | S-Partial Pressure | 38hex | i * | Partial Pressure | 13 | N.A. | REAL |
| Status i * | S-Partial Pressure | 38hex | i * | Status | 7 | N.A. | BYTE |

* The assignment of the Instance ID is a one-for-one mapping to the ordering of the members of the Input Assemblies in the above table. i.e., Partial Pressure 1 is the Partial Pressure from the S-Partial Pressure Object Instance 1. Therefore, the user must configure each instance of the S-Partial Pressure object for the desired Partial Pressure in the order they are to appear in the Input Assembly member list.
** Instance 0 corresponds to the Class Level


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-37.7 Object Limitations and Specific Definitions

This section describes limitations and specific definitions applicable to the listed objects of this device profile when these objects are applied in this type of device.

**Table 6-37.6 S-Device Supervisor**

| Attribute | Requirements |
|---|---|
| Device Type | Set to "RGA" for Residual Gas Analyzer Device |

The following table specifies the data attribute bit mapping for the **Device Exception Detail** bytes for this device. For more descriptive information, see the definition of the S-Device Supervisor Object Class.

Any Exception Bit not supported shall default to 0. Note that this profile allows for only one byte of manufacturer specific exception detail.

**Table 6-37.7 Attribute Bit Mapping**

| Data Component Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| RGA Device Exception Detail Size | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| RGA Device Exception Detail Byte 0 | 0 | 0 | 0 | 0 | 0 | Partial Pressure High | Partial Pressure Low | Reading Invalid |
| RGA Device Exception Detail Byte 1 | 0 | 0 | 0 | 0 | Filament 4 Fault | Filament 3 Fault | Filament 2 Fault | Filament 1 Fault |
| Manufacturer Exception Detail Size * | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Manufacturer Exception Detail * | 8 Bits defined by Manufacturer | | | | | | | |

* This example shows a one-byte manufacturer exception detail, but there may be zero or many bytes defined

**Table 6-37.8 S-Partial Pressure Object Class**

| Attribute | Additions / Limitation | Requirements |
|---|---|---|
| Max Instance | | Required |

### 6-37.7.1 S-Analog Sensor Object

#### 6-37.7.1.1 Class

An RGA device may be able to support an external Capacitance Manometer. However, an implementation may, or may not, include this connection. Therefore, devices that support this object shall support the Class Level services Create and Delete for the S-Analog Sensor object.

#### 6-37.7.1.2 Instance

Instance ID 1 (one) is used to interface to an external Capacitance Manometer.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 6-37.7.1.3 Attribute Definitions

The following attribute limitations apply:

**Table 6-37.9 Attribute Limitations**

| Attribute | Additions / Limitation | Requirements |
|---|---|---|
| Data Type | Supported Values = {REAL} | |
| Data Units | Supported Values = {Torr} | |
| Gain Data Type | Supported Values = {REAL} | |
| Gain | It is assumed that the external Capacitance Manometer is calibrated to yield a linear output from 0-10 Volts corresponding to Zero-Full Scale reading. | Shall be set to Full Scale (in Torr) of the externally connected Capacitance Manometer divided by 10 (ten). |

### 6-37.7.2 Discrete Input Point Object Instance

Instance ID 1 (one) is used to interface to an external Turbomolecular Pump. Specifically, this input is connected to the pump's At Speed indicator. Thus, the *Value* attribute indicates whether the pump is operating at the specified pumping speed.

0 = False / 1 = True.

### 6-37.7.3 Discrete Output Point Object Instance

#### 6-37.7.3.1 Object Instance Number Assignment

The following instance numbering limitations apply:

**Table 6-37.10 Instance Numbering Limitations**

| Instance | Description |
|---|---|
| 1 | Valve Open - connected externally to a Pump Isolation Valve. Used to command the valve to open or close.<br>0 = Close / 1 = Open |
| 2 | Turbo Pump On - connected externally to a Turbomolecular Pump. Used to command the pump on or off.<br>0 = Off / 1 = On |

## 6-37.8 Defining Device Configuration

Public access to the S-Device Supervisor and S-Partial Pressure Objects by the Message Router must be supported for configuration of this device type. There is no Parameter Object defined for access to the device type’s configuration parameters.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-38 DC Power Generator Device

**Device Type: 1F hex**

A DC Power Generator is a device that converts AC utility power to regulated DC power. It uses sensors to monitor the output, and a controller, which closes the loop with a setpoint such that the output is controlled to the setpoint. The setpoint value can be a voltage, current or power level.

## 6-38.1 Object Model

The Object Model in Figure 6-38.1 represents a DC Power Generator Device. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-38.1 Objects Present in a DC Power Generator Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Required | at least 5 Input and<br>5 Output |
| S-Device Supervisor | Required | 1 |
| S-Analog Sensor | Required | at least 1<br>as many as 6 |
| S-Analog Actuator | Required | 1 |
| S-Single Stage Controller | Required | 1 - 3 |
| Selection | Required | 1 |
| Register | Required | 2 |

## 6-38.2 Subclasses

Some of the object classes used by this profile define class and/or instance level subclasses. Each subclass defines a unique meaning for an overlapping range of attribute IDs and/or service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass in use, for a given object class or instance, by the device is identified by the value of its Subclass attribute. The following tables specify the subclass ID and corresponding object class and/or instance ID for this profile.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-38.2 S-Device Supervisor Object Subclass**

| Object Instance ID | Subclass Name | Subclass ID (Attribute 99 Value) | Required | Function | Restrictions |
|---|---|---|---|---|---|
| 1 | Power Generator | 01 | Required | Device Management | None |

**Table 6-38.3 S-Single Stage Controller Object Subclasses**

| Object Instance ID | Subclass Name | Subclass ID (Attribute 99 Value) | Required | Function | Restrictions |
|---|---|---|---|---|---|
| 1 | DC Generator | 02 | Conditional * | Power Regulation | Data Units = Watts or milliWatts |
| 2 | DC Generator | 02 | Conditional * | Voltage Regulation | Data Units = Volts |
| 3 | DC Generator | 02 | Conditional * | Current Regulation | Data Units = milliAmps |

* At least one of these instances shall be supported



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-38.1 Object Model for the DC Power Generator Device**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-38.3 How Objects Affect Behavior

**Table 6-38.4 Object Affect on Behavior**

| Object | Effect on behavior |
|---|---|
| Identity | Supports the Reset service. Upon receipt of a Reset Service Request of any Type, the Identity Object sends a Reset Service Request to the S-Device Supervisor. |
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines input/output and configuration data format. |
| S-Device Supervisor | Supports the Stop, Start, Reset, Abort, Recover and Perform_Diagnostic services for ALL Application Objects in the device and consolidates the Exception Conditions and Application Objects' Status.<br>This object behaves differently from the Identity Object in that the S-Device Supervisor object provides a single point of access to the Application Objects only; it does not effect the DeviceNet specific objects (i.e., Identity, DeviceNet, Connection, etc.). |
| S-Analog Sensor | Provides sensed measurements and feeds process variables to the S-Single Stage Controllers. |
| S-Single Stage Controller | The active instance feeds the control variable to the S-Analog Actuator whereby regulating the output as specified. |
| S-Analog Actuator | Operates the Power Generator output drive circuitry. |
| Selection | Activates an instance of the S-Single Stage Controller class and feeds it the Setpoint variable. |
| Register | Instance 1 - No effect.<br>Instance 2 - The transition of any interlock bit from 0 to 1 causes an internally generated Stop Service Request to the S-Device Supervisor object instance. Also, any interlock bit in the 1 state shall prevent the S-Device Supervisor object from the EXECUTING state. See Register section below. |

## 6-38.4 Defining Object Interfaces

**Table 6-38.5 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| S-Device Supervisor | Assembly or Message Router |
| S-Analog Sensor | Assembly or Message Router |
| S-Analog Actuator | Assembly or Message Router |
| S-Single Stage Controller | Assembly or Message Router |
| Selection | Assembly or Message Router |
| Register | Assembly or Message Router |

## 6-38.5 I/O Assembly Instances

The following table identifies the I/O assembly instances. The manufacturer must specify which Assembly instances are supported by the device.

Any element not supported must be set to zero (0).



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-38.6 I/O Assembly Instances**

| Instance | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Y (Default) | Output | 0 | Power Out 12 (low byte) | | | | | | | |
| | | | 1 | 0 | 0 | 0 | 0 | Power Out 12 (high nibble) | | | |
| | | | 2 - 3 | Power Ramp | | | | | | | |
| | | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 2 | Y (Default) | Input | 0 | Power In 12 (low byte) | | | | | | | |
| | | | 1 | 0 | 0 | 0 | 0 | Power In 12 (high nibble) | | | |
| | | | 2 | Voltage In 12 (low byte) | | | | | | | |
| | | | 3 | 0 | 0 | 0 | 0 | Voltage In 12 (high nibble) | | | |
| | | | 4 | Current In 12 (low byte) | | | | | | | |
| | | | 5 | 0 | 0 | 0 | 0 | Current In 12 (high nibble) | | | |
| | | | 6 - 7 | Power Ramp | | | | | | | |
| | | | 8 | Op Status | | | | | | | |
| 3 | Y | Output | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 4 | N | Output | 0 - 3 | Power Limit | | | | | | | |
| | | | 4 - 7 | Voltage Limit | | | | | | | |
| | | | 8 - 11 | Current Limit | | | | | | | |
| 5 | Y | Output | 0 - 1 | Regulation Select | | | | | | | |
| | | | 2 - 5 | Setpoint | | | | | | | |
| | | | 6 - 7 | Power Ramp | | | | | | | |
| | | | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 6 | N | Output | 0 - 1 | Regulation Select | | | | | | | |
| | | | 2 - 5 | Setpoint | | | | | | | |
| | | | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 7 | Y | Output | 0 - 3 | Power Out | | | | | | | |
| | | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 8 | Y | Output | 0 - 3 | Power Out | | | | | | | |
| | | | 4 - 5 | Power Ramp | | | | | | | |
| | | | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 9 | Y | Input | 0 | Ex Status | | | | | | | |
| | | | 1 - 4 | Power In | | | | | | | |
| | | | 5 - 8 | Voltage In | | | | | | | |
| | | | 9 - 12 | Current In | | | | | | | |
| 10 | N | Input | 0 | Ex Status | | | | | | | |
| | | | 1 - 4 | Energy In | | | | | | | |
| 11 | N | Input | 0 | Ex Status | | | | | | | |
| | | | 1 - 4 | Power In | | | | | | | |
| | | | 5 - 8 | Voltage In | | | | | | | |
| | | | 9 - 12 | Current In | | | | | | | |
| | | | 13 - 16 | Energy In | | | | | | | |
| 12 | N | Input | 0 | Ex Status | | | | | | | |
| | | | 1 - 2 | W Temp | | | | | | | |
| | | | 3 - 4 | D Temp | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## DC Power Generator Device, Type: $1F_{Hex}$

| Instance | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 13 | N | Output | 0 - 3 | Voltage Out | | | | | | | |
| | | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 14 | N | Output | 0 - 3 | Voltage Out | | | | | | | |
| | | | 4 - 5 | Voltage Ramp | | | | | | | |
| | | | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 15 | N | Output | 0 - 3 | Current Out | | | | | | | |
| | | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 16 | N | Output | 0 - 3 | Current Out | | | | | | | |
| | | | 4 - 5 | Current Ramp | | | | | | | |
| | | | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 17 | N | Input | 0 | Ex Status | | | | | | | |
| | | | 1 - 4 | Power In | | | | | | | |
| 18 | N | Input | 0 | Ex Status | | | | | | | |
| | | | 1 - 4 | Voltage In | | | | | | | |
| 19 | N | Input | 0 | Ex Status | | | | | | | |
| | | | 1 - 4 | Current In | | | | | | | |
| 20 | Y | Input | 0 | Ex Status | | | | | | | |
| 21 | Y | Input | 0 | Exception Detail Alarm 0 (size, common = 2) | | | | | | | |
| | | | 1 | Exception Detail Alarm 1 (common 0) | | | | | | | |
| | | | 2 | Exception Detail Alarm 2 (common 1) | | | | | | | |
| | | | 3 | Exception Detail Alarm 3 (size, device = 2) | | | | | | | |
| | | | 4 | Exception Detail Alarm 4 (device 0) | | | | | | | |
| | | | 5 | Exception Detail Alarm 5 (device 1) | | | | | | | |
| | | | 6 | Exception Detail Alarm 6 (size, manufacturer = 1) | | | | | | | |
| | | | 7 | Exception Detail Alarm 7 (manufacturer 0) | | | | | | | |
| 22 | Y | Input | 0 | Exception Detail Warning 0 (size, common = 2) | | | | | | | |
| | | | 1 | Exception Detail Warning 1 (common 0) | | | | | | | |
| | | | 2 | Exception Detail Warning 2 (common 1) | | | | | | | |
| | | | 3 | Exception Detail Warning 3 (size, device = 2) | | | | | | | |
| | | | 4 | Exception Detail Warning 4 (device 0) | | | | | | | |
| | | | 5 | Exception Detail Warning 5 (device 1) | | | | | | | |
| | | | 6 | Exception Detail Warning 6 (size, manufacturer = 1) | | | | | | | |
| | | | 7 | Exception Detail Warning 7 (manufacturer 0) | | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**DC Power Generator Device, Type: $1F_{Hex}$**

| Instance | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 23 | N | Input | 0 | Ex Status | | | | | | | |
| | | | 1 | Exception Detail Alarm 0 (size, common = 2) | | | | | | | |
| | | | 2 | Exception Detail Alarm 1 (common 0) | | | | | | | |
| | | | 3 | Exception Detail Alarm 2 (common 1) | | | | | | | |
| | | | 4 | Exception Detail Alarm 3 (size, device = 2) | | | | | | | |
| | | | 5 | Exception Detail Alarm 4 (device 0) | | | | | | | |
| | | | 6 | Exception Detail Alarm 5 (device 1) | | | | | | | |
| | | | 7 | Exception Detail Alarm 6 (size, manufacturer = 1) | | | | | | | |
| | | | 8 | Exception Detail Alarm 7 (manufacturer 0) | | | | | | | |
| | | | 9 | Exception Detail Warning 0 (size, common = 2) | | | | | | | |
| | | | 10 | Exception Detail Warning 1 (common 0) | | | | | | | |
| | | | 11 | Exception Detail Warning 2 (common 1) | | | | | | | |
| | | | 12 | Exception Detail Warning 3 (size, device = 2) | | | | | | | |
| | | | 13 | Exception Detail Warning 4 (device 0) | | | | | | | |
| | | | 14 | Exception Detail Warning 5 (device 1) | | | | | | | |
| | | | 15 | Exception Detail Warning 6 (size, manufacturer = 1) | | | | | | | |
| | | | 16 | Exception Detail Warning 7 (manufacturer 0) | | | | | | | |
| 24 | N | Input | 0 | Op Status | | | | | | | |
| 25 | N | Input | 0 | Interlock Status (low byte) | | | | | | | |
| | | | 1 | Interlock Status (high byte) | | | | | | | |
| 26 | N | Input | 0 | Ex Status | | | | | | | |
| | | | 1 | Op Status | | | | | | | |
| | | | 2 | Interlock Status (low byte) | | | | | | | |
| | | | 3 | Interlock Status (high bytes) | | | | | | | |

### 6-38.5.1 I/O Assembly Instance Component Formats

All of the elemental components listed above follow standard CIP Data Encoding as specified in Appendix C. The encoding order for all data is Low Byte first. All members are LSB aligned and end on Byte boundaries. That is, upper bits shall be stuffed with 0 as needed to fill to the next byte.

### 6-38.5.2 I/O Assembly Instance Component Mapping

Each of the *S-Analog Sensor*, *S-Analog Actuator* and *S-Single Stage Controller* object definitions specify a behavior that modifies the *Data Type* of certain attributes based upon the first valid I/O connection established to an Assembly Object instance. See these object definitions for more detail.

The following table indicates the I/O assembly Data attribute mapping for this device.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-38.7 I/O Assembly Data Mapping**

| Data Component | Object Name | Class ID | Instance ID | Attribute Name | Attribute ID | Data Type |
|---|---|---|---|---|---|---|
| Power Out 12 | S-Single Stage Controller | $33_{hex}$ | 1 | Setpoint | 6 | UINT |
| Power In 12 | S-Analog Sensor | $31_{hex}$ | 1 | Value | 6 | UINT |
| Voltage In 12 | S-Analog Sensor | $31_{hex}$ | 2 | Value | 6 | UINT |
| Current In 12 | S-Analog Sensor | $31_{hex}$ | 3 | Value | 6 | UINT |
| Power Ramp | S-Single Stage Controller | $33_{hex}$ | 1 | Ramp Rate Increment | 96 | UINT |
| Voltage Ramp | S-Single Stage Controller | $33_{hex}$ | 2 | Ramp Rate Increment | 96 | UINT |
| Current Ramp | S-Single Stage Controller | $33_{hex}$ | 3 | Ramp Rate Increment | 96 | UINT |
| On | S-Device Supervisor | $30_{hex}$ | 1 | Output Power Enable | 96 | BOOL |
| Op Status | Register | $07_{hex}$ | 1 | Data | 4 | ARRAY of 8 BITS |
| Power Limit | S-Single Stage Controller | $33_{hex}$ | 1 | Output Limit | 86 | DINT |
| Voltage Limit | S-Single Stage Controller | $33_{hex}$ | 2 | Output Limit | 86 | DINT |
| Current Limit | S-Single Stage Controller | $33_{hex}$ | 3 | Output Limit | 86 | DINT |
| Regulation Select | Selection | $2E_{hex}$ | 1 | destination_used | 14 | UINT |
| Setpoint | Selection | $2E_{hex}$ | 1 | input_data_value | 15 | DINT |
| Power Out | S-Single Stage Controller | $33_{hex}$ | 1 | Setpoint | 6 | DINT |
| Voltage Out | S-Single Stage Controller | $33_{hex}$ | 2 | Setpoint | 6 | DINT |
| Current Out | S-Single Stage Controller | $33_{hex}$ | 3 | Setpoint | 6 | DINT |
| Power In | S-Analog Sensor | $31_{hex}$ | 1 | Value | 6 | DINT |
| Voltage In | S-Analog Sensor | $31_{hex}$ | 2 | Value | 6 | DINT |
| Current In | S-Analog Sensor | $31_{hex}$ | 3 | Value | 6 | DINT |
| Energy In | S-Analog Sensor | $31_{hex}$ | 4 | Value | 6 | DINT |
| W Temp | S-Analog Sensor | $31_{hex}$ | 5 | Value | 6 | INT |
| D Temp | S-Analog Sensor | $31_{hex}$ | 6 | Value | 6 | INT |
| Interlock Status | Register | $07_{hex}$ | 2 | Data | 4 | ARRAY of 16 BITS |
| Ex Status | S-Device Supervisor | $30_{hex}$ | 1 | Exception Status | 12 | BYTE |
| Exception Alarm | S-Device Supervisor | $30_{hex}$ | 1 | Exception Detail Alarm | 13 | STRUCT |
| Exception Warning | S-Device Supervisor | $30_{hex}$ | 1 | Exception Detail Warning | 14 | STRUCT |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-38.6 Object Limitations and Specific Definitions

This section describes limitations and specific definitions applicable to the listed objects of this device profile when these objects are applied in this type of device.

### 6-38.6.1 S-Device Supervisor

The following attribute limitations apply:

**Table 6-38.8 Device Type Attribute Limitation**

| Attribute | Requirements |
|---|---|
| Device Type | Set to "DCG" for DC Power Generator Device |

The following table specifies the data attribute bit mapping for the **Device Exception Detail** bytes for this device. For more descriptive information, see the definition of the S-Device Supervisor Object Class.

Any Exception Bit not supported shall default to 0. Note that this profile allows for only one byte of manufacturer specific exception detail.

**Table 6-38.9 Data Attribute Bit Mapping**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Device Exception Detail Size | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Device Exception Detail Byte 0 | Reserved 0 | Input Power V Low | Input Power V High * | End of Target Life * | Cooling Failure | Device Temp High | Interlock Open | Control Circuit Fault |
| Byte 1 | Reserved 0 | Reserved 0 | Reserved 0 | Reserved 0 | Reserved 0 | Arc Limit Exceeded * | Process V Low | Regulation |
| Manufacturer Exception Detail Size | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Manufacturer Exception Detail Byte 0 | 8 Bits defined by Manufacturer | | | | | | | |

* Only used in the Warning Exception Detail, this bit is always = 0 in the Alarm Exception Detail



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-38.6.2 S-Analog Sensor Object Instances

**Table 6-38.10 Object Instances**

| Object Instance ID | Function | Required | Limitations and Specific Definitions |
|---|---|---|---|
| 1 | Output Power | Required | Supported Data Type and Data Units pairs:<br>DINT / Watts;<br>DINT / milliWatts<br>UINT / % * |
| 2 | Output Voltage | Optional | Default Full Scale = 1.0 KV<br>Supported Data Type and Data Units pairs:<br>DINT / Volts (rms);<br>UINT / % * |
| 3 | Output Current | Optional | Supported Data Type and Data Units pairs:<br>DINT / milliAmps (rms);<br>UINT / % * |
| 4 | Delivered Energy since Output Power was last turned On | Optional | DINT / Joules |
| 5 | Cooling Water Temp | Optional | INT / °C<br>Resolution = 1 Degree |
| 6 | Device Temp | Optional | INT / °C<br>Resolution = 1 Degree |

* Range = 0 - 4095 corresponding to 0 - 100%

### 6-38.6.3 S-Analog Actuator Object Instances

**Table 6-38.11 Object Instances**

| Object Instance ID | Function | Required | Limitations and Specific Definitions |
|---|---|---|---|
| 1 | Output Power | Required | Supported Data Type and Data Units pairs:<br>DINT / Watts;<br>DINT / milliWatts<br>UINT / % * |

* Range = 0 - 4095 corresponding to 0 - 100%


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-38.6.4 S-Single Stage Controller Object Instances

**Table 6-38.12 Object Instances**

| Object Instance ID | Function | Required | Limitations and Specific Definitions |
|---|---|---|---|
| 1 | Output Power Regulation | Optional | Supported Data Type and Data Units pairs:<br>DINT / Watts;<br>DINT / milliWatts;<br>UINT / % * |
| 2 | Output Voltage Regulation | Optional | Supported Data Type and Data Units pairs:<br>DINT / Volts (rms);<br>UINT / % * |
| 3 | Output Current Regulation | Optional | Supported Data Type and Data Units pairs:<br>DINT / milliAmps (rms);<br>UINT / % * |
| * Range = 0 - 4095 corresponding to 0 - 100% | | | |

#### 6-38.6.4.1 Behavior

Of instances 1, 2 and 3, only one is active at any given time. Which instance is active is determined by the value of the *destination_used* attribute of the Selection object instance. In typical operation, the *destination_used* attribute is mapped to, and hence modified by, the Regulation Select member of an Output Assembly Instance, which in turn, is referenced by the *Consume Path* of the first valid I/O connection established by the device.

#### 6-38.6.4.2 Selection Object, Instance 1

**Table 6-38.13 Selection Object – Instance 1 Attribute Limitations**

| Attribute | Limitation (assigned values are shown in hex format) |
|---|---|
| max_destination | 03 |
| number_of_destinations | 03 |
| destination_list | 06 / 20 / 33 / 24 / 01 / 30 / 06 — S-Single Stage Controller, Instance 1, Setpoint<br>06 / 20 / 33 / 24 / 02 / 30 / 06 — S-Single Stage Controller, Instance 2, Setpoint<br>06 / 20 / 33 / 24 / 03 / 30 / 06 — S-Single Stage Controller, Instance 3, Setpoint<br>Access Rule = Get Only |
| max_sources | 00 |
| number_of_sources | 00 |
| source_list | Not Supported |
| source_used | 00 |
| algorithm_used | 04 - Programmable Data Flow<br>Access Rule = Get Only |
| object_source_list | 00<br>Access Rule = Get Only |
| destination_used | Range = 0 - 3 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-38.6.5 Register Object

#### 6-38.6.5.1 Register Object, Instance 1 (Input) Operational Status

**Table 6-38.14 Data Attribute (8 Bits)**

| Bit | Description |
|---|---|
| 0 | Power On Status (1=on / 0=off) |
| 1 | Setpoint Status (1=at setpoint / 0=not) |
| 2 | Temp Status (1=OK / 0=overtemp) |
| 3 | Arc Detected (1=on / 0=off) |
| 4 | Interlock Status Flag (1=OK / 0=open) |
| 5 | Reserved for Future Use (0) |
| 6 | Reserved for Future Use (0) |
| 7 | Reserved for Future Use (0) |

#### 6-38.6.5.2 Register Object, Instance 2 (Input) Interlock Status

**Table 6-38.15 Data Attribute (16 Bits)**

| Bit | Description |
|---|---|
| 0 | Device Not Ready |
| 1 | Communications Fault |
| 2 | Device State Fault |
| 3 | Device Not Configured |
| 4 | External |
| 5 | Head Cable |
| 6 | Head Cover |
| 7 | High-Voltage Cable |
| 8 | Bias Supply |
| 9 | DC Bus Voltage |
| 10 | Fan Failure |
| 11 | Magnet Temp |
| 12 | Power  Monitor |
| 13 | Cover |
| 14 | Reserved for Future Use (0) |
| 15 | Reserved for Future Use (0) |
| 1 = Interlocked / 0 = Not Interlocked | |

#### 6-38.6.5.3 Behavior

The transition of any interlock bit from 0 to 1 shall cause an internally generated Stop Service Request to the S-Device Supervisor object instance thereby causing a transition to the IDLE state and disabling the output power of the device.  Also, if any interlock bit has a value of 1, the S-Device Supervisor object instance shall not transition to the EXECUTING State: (a) a Start Service Request shall return a Device_State_Conflict error, and (b) a Receipt of First Valid I/O Data event shall not cause a state transition.

### 6-38.7 Defining Device Configuration

Public access to the S-Device Supervisor and the application objects by the Message Router must be supported for configuration of this device type.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-39 RF Power Generator Device

**Device Type: 20 hex**

An RF Power Generator is a device that converts AC utility power to regulated RF power. It uses sensors to monitor the output, and a controller, which closes the loop with a setpoint such that the output is controlled to the setpoint. The setpoint value can be a forward or delivered power level. The output frequency can be fixed, set with a user setpoint, or automatically tunable by a controller, which attempts to minimize reflected power.

## 6-39.1 Object Model

The Object Model in Figure 6-39.1 represents an RF Power Generator Device. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-39.1 Objects Present in a RF Power Generator Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Assembly | Required | at least<br>4 Input and<br>2 Output |
| S-Device Supervisor | Required | 1 |
| S-Analog Sensor | Required | at least 3<br>as many as 15 |
| S-Analog Actuator | Required | at least 2<br>as many as 4 |
| S-Single Stage Controller | Optional | 1 - 3 |
| Selection | Required | 1 |
| Register | Required | 2 |

## 6-39.2 Subclasses

Some of the object classes used by this profile define class and/or instance level subclasses. Each subclass defines a unique meaning for an overlapping range of attribute IDs and/or service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass in use, for a given object class or instance, by the device is identified by the value of its Subclass attribute. The following tables specify the subclass ID and corresponding object class and/or instance ID for this profile.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-39.2 S-Device Supervisor Object Subclass**

| Object Instance ID | Subclass Name | Subclass ID (Attribute 99 Value) | Required | Function | Restrictions |
|---|---|---|---|---|---|
| 1 | Power Generator | 01 | Required | Device Management | None |

**Table 6-39.3 S-Single Stage Controller Object Subclasses**

| Object Instance ID | Subclass Name | Subclass ID (Attribute 99 Value) | Required | Function | Restrictions |
|---|---|---|---|---|---|
| 1 | RF Generator | 03 | Optional | Forward Power Regulation | Data Units = Watts or milliWatts |
| 2 | RF Generator | 03 | Optional | Delivered Power Regulation | Data Units = Watts |
| 3 | Frequency Control | 04 | Optional | Frequency Regulation | Data Units = Hertz |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-39.1 Object Model for the RF Power Generator Device**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-39.3 How Objects Affect Behavior

**Table 6-39.4 Object Affect on Behavior**

| Object | Effect on behavior |
|---|---|
| Identity | Supports the Reset service. Upon receipt of a *Reset* Service Request of any *Type*, the Identity Object sends a *Reset* Service Request to the S-Device Supervisor. |
| CIP Required Objects | See Section 6-2.2 for details. |
| Assembly | Defines input/output and configuration data format |
| S-Device Supervisor | Supports the Stop, Start, Reset, Abort, Recover and Perform_Diagnostic services for ALL Application Objects in the device and consolidates the Exception Conditions and Application Objects' Status.<br>This object behaves differently from the Identity Object in that the S-Device Supervisor object provides a single point of access to the Application Objects only; it does not effect the DeviceNet specific objects (i.e., Identity, DeviceNet, Connection, etc.). |
| S-Analog Sensor | Provides sensed measurements and feeds process variables to the S-Single Stage Controllers |
| S-Single Stage Controller | The active instance feeds the control variable to the S-Analog Actuator thereby regulating the output as specified |
| S-Analog Actuator | Operates the Power Generator output drive circuitry |
| Selection | Activates an instance of the S-Single Stage Controller class and feeds it the Setpoint variable |
| Register | Instance 1 - No effect.<br>Instance 2 - The transition of any interlock bit from 0 to 1 causes an internally generated Stop Service Request to the S-Device Supervisor object instance. |

## 6-39.4 Defining Object Interfaces

**Table 6-39.5 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details. |
| Assembly | I/O Connection or Message Router |
| S-Device Supervisor | Assembly or Message Router |
| S-Analog Sensor | Assembly or Message Router |
| S-Analog Actuator | Assembly or Message Router |
| S-Single Stage Controller | Assembly or Message Router |
| Selection | Assembly or Message Router |
| Register | Assembly or Message Router |

## 6-39.5 I/O Assembly Instances

The following table identifies the I/O assembly instances. The manufacturer must specify which Assembly instances are supported by the device.

Any element not supported must be set to zero (0).



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-39.6 I/O Assembly Instances**

| Instance | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Y (Default) | Output | 0 | Forward Power Out 12 (low byte) | | | | | | | |
| | | | 1 | 0 | 0 | 0 | 0 | Power Out 12 (high nibble) | | | |
| | | | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 2 | Y (Default) | Input | 0 | Forward Power In 12 (low byte) | | | | | | | |
| | | | 1 | 0 | 0 | 0 | 0 | Forward Power In 12 (high nibble) | | | |
| | | | 2 | Reflected Power In 12 (low byte) | | | | | | | |
| | | | 3 | 0 | 0 | 0 | 0 | Reflected Voltage In 12 (high nibble) | | | |
| | | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | 8 | Op Status | | | | | | | |
| 3 | Y | Output | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 4 | N | Output | 0 - 3 | Forward Power Limit | | | | | | | |
| | | | 4 - 7 | Delivered Power Limit | | | | | | | |
| | | | 8 - 11 | Frequency Limit Low | | | | | | | |
| | | | 12 - 15 | Frequency Limit High | | | | | | | |
| 5 | N | Output | 0 - 1 | Regulation Select | | | | | | | |
| | | | 2 - 5 | Setpoint | | | | | | | |
| | | | 6 - 7 | Power Ramp | | | | | | | |
| | | | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 6 | N | Output | 0 - 1 | Regulation Select | | | | | | | |
| | | | 2 - 5 | Setpoint | | | | | | | |
| | | | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 7 | N | Output | 0 - 3 | Forward Power Out | | | | | | | |
| | | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 8 | N | Output | 0 - 3 | Forward Power Out | | | | | | | |
| | | | 4 - 5 | Power Ramp | | | | | | | |
| | | | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 9 | N | Output | 0 - 3 | Delivered Power Out | | | | | | | |
| | | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 10 | N | Output | 0 - 3 | Delivered Power Out | | | | | | | |
| | | | 4 - 5 | Power Ramp | | | | | | | |
| | | | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | On |
| 11 | N | Input | 0 | Ex Status | | | | | | | |
| | | | 1 - 4 | Forward Power In | | | | | | | |
| | | | 5 - 8 | Voltage In | | | | | | | |
| | | | 9 - 12 | Current In | | | | | | | |
| 12 | N | Input | 0 | Ex Status | | | | | | | |
| | | | 1 - 4 | Energy In | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## RF Power Generator Device, Type: $20_{Hex}$

| Instance | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 13 | N | Input | 0 | Ex Status | | | | | | | |
| | | | 1 - 4 | Forward Power In | | | | | | | |
| | | | 5 - 8 | Voltage In | | | | | | | |
| | | | 9 - 12 | Current In | | | | | | | |
| | | | 13 - 16 | Energy In | | | | | | | |
| 14 | N | Input | 0 | Ex Status | | | | | | | |
| | | | 1 - 4 | Forward Power In | | | | | | | |
| | | | 5 - 8 | Reflected Power In | | | | | | | |
| | | | 9 - 12 | Voltage In | | | | | | | |
| | | | 13 - 16 | Current In | | | | | | | |
| 15 | N | Input | 0 | Ex Status | | | | | | | |
| | | | 1 - 4 | Delivered Power In | | | | | | | |
| 16 | N | Input | 0 | Ex Status | | | | | | | |
| | | | 1 - 2 | W Temp | | | | | | | |
| | | | 3 - 4 | D Temp | | | | | | | |
| 17 | Y | Input | 0 | Ex Status | | | | | | | |
| 18 | Y | Input | 0 | Exception Detail Alarm 0 (size, common = 2) | | | | | | | |
| | | | 1 | Exception Detail Alarm 1 (common 0) | | | | | | | |
| | | | 2 | Exception Detail Alarm 2 (common 1) | | | | | | | |
| | | | 3 | Exception Detail Alarm 3 (size, device = 2) | | | | | | | |
| | | | 4 | Exception Detail Alarm 4 (device 0) | | | | | | | |
| | | | 5 | Exception Detail Alarm 5 (device 1) | | | | | | | |
| | | | 6 | Exception Detail Alarm 6 (size, manufacturer = 1) | | | | | | | |
| | | | 7 | Exception Detail Alarm 7 (manufacturer 0) | | | | | | | |
| 19 | Y | Input | 0 | Exception Detail Warning 0 (size, common = 2) | | | | | | | |
| | | | 1 | Exception Detail Warning 1 (common 0) | | | | | | | |
| | | | 2 | Exception Detail Warning 2 (common 1) | | | | | | | |
| | | | 3 | Exception Detail Warning 3 (size, device = 2) | | | | | | | |
| | | | 4 | Exception Detail Warning 4 (device 0) | | | | | | | |
| | | | 5 | Exception Detail Warning 5 (device 1) | | | | | | | |
| | | | 6 | Exception Detail Warning 6 (size, manufacturer = 1) | | | | | | | |
| | | | 7 | Exception Detail Warning 7 (manufacturer 0) | | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**RF Power Generator Device, Type: $20_{Hex}$**

| Instance | Required | Type | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 20 | N | Input | 0 | Ex Status | | | | | | | |
| | | | 1 | Exception Detail Alarm 0 (size, common = 2) | | | | | | | |
| | | | 2 | Exception Detail Alarm 1 (common 0) | | | | | | | |
| | | | 3 | Exception Detail Alarm 2 (common 1) | | | | | | | |
| | | | 4 | Exception Detail Alarm 3 (size, device = 2) | | | | | | | |
| | | | 5 | Exception Detail Alarm 4 (device 0) | | | | | | | |
| | | | 6 | Exception Detail Alarm 5 (device 1) | | | | | | | |
| | | | 7 | Exception Detail Alarm 6 (size, manufacturer = 1) | | | | | | | |
| | | | 8 | Exception Detail Alarm 7 (manufacturer 0) | | | | | | | |
| | | | 9 | Exception Detail Warning 0 (size, common = 2) | | | | | | | |
| | | | 10 | Exception Detail Warning 1 (common 0) | | | | | | | |
| | | | 11 | Exception Detail Warning 2 (common 1) | | | | | | | |
| | | | 12 | Exception Detail Warning 3 (size, device = 2) | | | | | | | |
| | | | 13 | Exception Detail Warning 4 (device 0) | | | | | | | |
| | | | 14 | Exception Detail Warning 5 (device 1) | | | | | | | |
| | | | 15 | Exception Detail Warning 6 (size, manufacturer = 1) | | | | | | | |
| | | | 16 | Exception Detail Warning 7 (manufacturer 0) | | | | | | | |
| 21 | N | Input | 0 | Op Status | | | | | | | |
| 22 | N | Input | 0 | Interlock Status (low byte) | | | | | | | |
| | | | 1 | 0 | 0 | Interlock Status (high 6 bits) | | | | | |
| 23 | N | Input | 0 | Ex Status | | | | | | | |
| | | | 1 | Op Status | | | | | | | |
| | | | 2 | Interlock Status (low byte) | | | | | | | |
| | | | 3 | 0 | 0 | Interlock Status (high 6 bits) | | | | | |

### 6-39.5.1 I/O Assembly Instance Component Formats

All of the elemental components listed above follow standard CIP Data Encoding as specified in Appendix C. The encoding order for all data is Low Byte first. All members are LSB aligned and end on Byte boundaries. That is, upper bits are stuffed with 0 as needed to fill to the next byte.

### 6-39.5.2 I/O Assembly Instance Component Mapping

Each of the *S-Analog Sensor*, *S-Analog Actuator* and *S-Single Stage Controller* object definitions specifies a behavior that modifies the *Data Type* of certain attributes based upon the first valid I/O connection established to an Assembly Object instance. See these object definitions for more detail.

The following table indicates the I/O assembly Data attribute mapping for this device.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-39.7 I/O Assembly Data Mapping**

| Data Component | Object Name | Class ID | Instance ID | Attribute Name | Attribute ID | Data Type |
|---|---|---|---|---|---|---|
| Forward Power Out 12 | S-Single Stage Controller | $33_{hex}$ | 1 | Setpoint | 6 | UINT |
| Forward Power In 12 | S-Analog Sensor | $31_{hex}$ | 1 | Value | 6 | UINT |
| Reflected Power In 12 | S-Analog Sensor | $31_{hex}$ | 2 | Value | 6 | UINT |
| Power Ramp | S-Single Stage Controller | $33_{hex}$ | 1 | Ramp Rate Increment | 96 | UINT |
| On | S-Device Supervisor | $30_{hex}$ | 1 | Output Power Enable | 96 | BOOL |
| Op Status | Register | $07_{hex}$ | 1 | Data | 4 | ARRAY of 8 BITS |
| Forward Power Limit | S-Single Stage Controller | $33_{hex}$ | 1 | Output Limit | 86 | DINT |
| Delivered Power Limit | S-Single Stage Controller | $33_{hex}$ | 2 | Output Limit | 86 | DINT |
| Frequency Limit Low | S-Single Stage Controller | $33_{hex}$ | 3 | Frequency Limit Low | 95 | DINT |
| Frequency Limit High | S-Single Stage Controller | $33_{hex}$ | 3 | Frequency Limit High | 96 | DINT |
| Filament On/Off | S-Device Supervisor | $30_{hex}$ | 1 | Filament Power Enable | 95 | BOOL |
| Regulation Select | Selection | $2E_{hex}$ | 1 | destination_used | 14 | UINT |
| Setpoint | Selection | $2E_{hex}$ | 1 | input_data_value | 15 | DINT |
| Forward Power Out | S-Single Stage Controller | $33_{hex}$ | 1 | Setpoint | 6 | DINT |
| Forward Power In | S-Analog Sensor | $31_{hex}$ | 1 | Value | 6 | DINT |
| Reflected Power In | S-Analog Sensor | $31_{hex}$ | 2 | Value | 6 | DINT |
| Voltage In | S-Analog Sensor | $31_{hex}$ | 4 | Value | 6 | DINT |
| Current In | S-Analog Sensor | $31_{hex}$ | 5 | Value | 6 | DINT |
| Energy In | S-Analog Sensor | $31_{hex}$ | 7 | Value | 6 | DINT |
| Delivered Power Out | S-Single Stage Controller | $33_{hex}$ | 2 | Setpoint | 6 | DINT |
| Delivered Power In | S-Analog Sensor | $31_{hex}$ | 3 | Value | 6 | DINT |
| W Temp | S-Analog Sensor | $31_{hex}$ | 8 | Value | 6 | INT |
| D Temp | S-Analog Sensor | $31_{hex}$ | 9 | Value | 6 | INT |
| Interlock | Register | $07_{hex}$ | 2 | Data | 4 | ARRAY of 16 BITS |
| Ex Status | S-Device Supervisor | $30_{hex}$ | 1 | Exception Status | 12 | BYTE |
| Ex Alarm | S-Device Supervisor | $30_{hex}$ | 1 | Exception Detail Alarm | 13 | STRUCT |
| Ex Warn | S-Device Supervisor | $30_{hex}$ | 1 | Exception Detail Warning | 14 | STRUCT |

## 6-39.6 Object Limitations and Specific Definitions

This section describes limitations and specific definitions applicable to the listed objects of this device profile when these objects are applied in this type of device.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-39.6.1 S-Device Supervisor

The following attribute limitations apply:

**Table 6-39.8 Device Type Attribute Limitation**

| Attribute | Requirements |
|---|---|
| Device Type | Shall be set to "RFG" for RF Power Generator Device |

The following table specifies the data attribute bit mapping for the **Device Exception Detail** bytes for this device. For more descriptive information, see the definition of the S-Device Supervisor Object Class.

Any Exception Bit not supported shall default to 0. Note that this profile allows for only one byte of manufacturer specific exception detail.

**Table 6-39.9 Data Attribute Bit Mapping**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Device Exception Detail Size | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Device Exception Detail Byte 0 | Magnetron Assembly Fault | Power Detector Fault | Contactor Closed | Fan Failure | Cooling Failure | Device Temp High | Interlock Open | Control Circuit Fault |
| Byte 1 | Input Power V Low | Input Power V High * | Output I Low | Output V Low | Output V High | Reflected P High | Output P High | Regulation |
| Byte 2 | IGBT Driver | Xfmr I High | Magnet V High | Bus V High | Bus V Low | Bus Shorted | DC I High | DC V Error |
| Byte 3 | Reserved 0 | Reserved 0 | End of Target Life * | Dew Conden-sation | Non-Zero Idle Output I | Non-Zero Idle Output V | Anode I High | Anode V High |
| Manufacturer Exception Detail Size | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Manufacturer Exception Detail Byte 0 | 8 Bits defined by Manufacturer | | | | | | | |

* Only used in the Warning Exception Detail, this bit is always = 0 in the Alarm Exception Detail.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-39.6.2 S-Analog Sensor Object Instances

**Table 6-39.10 Object Instances**

| Object Instance ID | Function | Required | Limitations and Specific Definitions |
|---|---|---|---|
| 1 | Forward Power | Required | Supported Data Type and Data Units pairs:<br>DINT / Watts;<br>DINT / milliWatts<br>UINT / % * |
| 2 | Reflected Power | Required | Supported Data Type and Data Units pairs:<br>DINT / Watts (rms);<br>DINT / milliWatts (rms)<br>UINT / % * |
| 3 | Delivered Power | Optional | Supported Data Type and Data Units pairs:<br>DINT / Watts (rms);<br>DINT / milliWatts (rms)<br>UINT / % * |
| 4 | Output Voltage | Optional | Supported Data Type and Data Units pairs:<br>DINT / Volts (rms);<br>UINT / % * |
| 5 | Output Current | Optional | Supported Data Type and Data Units pairs:<br>DINT / milliAmps (rms);<br>UINT / % * |
| 6 | Output Frequency | Required | DINT / Hertz |
| 7 | Delivered Energy since Output Power was last turned On | Optional | DINT / Joule |
| 8 | Cooling Water Temp | Optional | INT / °C<br>Resolution = 1 Degree |
| 9 | Device Temp | Optional | INT / °C<br>Resolution = 1 Degree |
| 10 | Plate Voltage | Optional | UINT / Volts |
| 11 | Plate Current | Optional | UINT / milliAmps |
| 12 | Grid Voltage | Optional | UINT / Volts |
| 13 | Grid Current | Optional | UINT / milliAmps |
| 14 | Driver Forward Power | Optional | UINT / Watts |
| 15 | Driver Reflected Power | Optional | UINT / Watts |
| * Range = 0 - 4095 corresponding to 0 - 100% | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-39.6.3 S-Analog Actuator Object Instances

**Table 6-39.11 Object Instances**

| Object Instance ID | Function | Required | Limitations and Specific Definitions |
|---|---|---|---|
| 1 | Output Power | Required | Supported Data Type and Data Units pairs:<br>DINT / Watts;<br>DINT / milliWatts<br>UINT / % * |
| 2 | Output Frequency | Required | Supported Data Type and Data Units pairs:<br>DINT / Hertz;<br>UINT / % * |
| 3 | Variable Tuning Element 1 | Optional | USINT / % of Max |
| 4 | Variable Tuning Element 2 | Optional | USINT / % of Max |
| * Range = 0 - 4095 corresponding to 0 - 100% | | | |

#### 6-39.6.3.1 S-Single Stage Controller Object Instances

**Table 6-39.12 Object Instances**

| Object Instance ID | Function | Required | Limitations and Specific Definitions |
|---|---|---|---|
| 1 | Output Forward Power Regulation | Optional | Supported Data Type and Data Units pairs:<br>DINT / Watts;<br>DINT / milliWatts<br>UINT / % * |
| 2 | Output Delivered Power Regulation | Optional | Supported Data Type and Data Units pairs:<br>DINT / Watts;<br>DINT / milliWatts<br>UINT / % * |
| 3 | Output Frequency Regulation | Optional | Supported Data Type and Data Units pairs:<br>DINT / Hertz;<br>UINT / % * |
| * Range = 0 - 4095 corresponding to 0 - 100% | | | |

#### 6-39.6.3.2 Behavior

Of instances 1 and 2, only one is active at any given time. Which instance is active is determined by the value of the *destination_used* attribute of the Selection object instance. In typical operation, the *destination_used* attribute is mapped to, and hence modified by, the Regulation Select member of an Output Assembly Instance, which in turn, is referenced by the *Consume Path* of the first valid I/O connection established by the device.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-39.6.4 Selection Object, Instance 1

**Table 6-39.13 Object Instances**

| Attribute | Limitation (assigned values are shown in hex format) |
|---|---|
| Max_destination | 02 |
| Number_of_destinations | 02 |
| Destination_list | 06 / 20 / 33 / 24 / 01 / 30 / 06 — S-Single Stage Controller, Instance 1, Setpoint<br>06 / 20 / 33 / 24 / 02 / 30 / 06 — S-Single Stage Controller, Instance 2, Setpoint<br>Access Rule = Get Only |
| Max_sources | 00 |
| Number_of_sources | 00 |
| Source_list | Not Supported |
| Source_used | 00 |
| Algorithm_used | 04 - Programmable Data Flow. Access Rule = Get Only |
| Object_source_list | 00 Access Rule = Get Only |
| Destination_used | Range = 0 - 2 |

### 6-39.6.5 Register Object

#### 6-39.6.5.1 Register Object, Instance 1 Operational Status

**Table 6-39.14 Object Instances**

| Bit | Description |
|---|---|
| 0 | Power On Status (1=on / 0=off) |
| 1 | Setpoint Status (1=at setpoint / 0=not) |
| 2 | Temp Status (1=OK / 0=overtemp) |
| 3 | Reserved (0) |
| 4 | Interlock Status (1=OK / 0=open) |
| 5 | Reserved for Future Use (0) |
| 6 | Reserved for Future Use (0) |
| 7 | Reserved for Future Use (0) |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-39.6.5.2 Register Object, Instance 2 Interlock Status

**Table 6-39.15 Object Instances**

| Bit | Description |
|---|---|
| 0 | Device Not Ready |
| 1 | Communications Fault |
| 2 | Device State Fault |
| 3 | Device Not Configured |
| 4 | External |
| 5 | Head Cable |
| 6 | Head Cover |
| 7 | High-Voltage Cable |
| 8 | Bias Supply |
| 9 | DC Bus Voltage |
| 10 | Fan Failure |
| 11 | Magnet Temp |
| 12 | Power Monitor |
| 13 | Cover |
| 14 | Reserved for Future Use (0) |
| 15 | Reserved for Future Use (0) |
| 1 = Interlocked / 0 = Not Interlocked | |

### 6-39.6.5.3 Behavior

The transition of any interlock bit from 0 to 1 shall cause an internally generated Stop Service Request to the S-Device Supervisor object instance thereby causing a transition to the IDLE state and disabling the output power of the device. Also, if any interlock bit has a value of 1, the S-Device Supervisor object instance shall not transition to the EXECUTING State: (a) a Start Service Request shall return a Device_State_Conflict error, and (b) a Receipt of First Valid I/O Data event shall not cause a state transition.

## 6-39.7 Defining Device Configuration

Public access to the S-Device Supervisor and the application objects by the Message Router must be supported for configuration of this device type.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-40 Fluid Flow Controller Device

**Device Type: 24 hex**

A Fluid Flow Controller (FFC) is a device that measures and controls the flow rate of a fluid (liquid or gas). It contains three principle components: a flow rate sensor which can be a virtual sensor based on other sensors, for example pressure; a fluid flow metering valve which can be actuated by one of a variety of actuator types, including, for example, solenoid or a stepper motor; and, a controller which closes the loop by receiving a setpoint and driving the actuator such that the fluid flow rate is controlled to the setpoint.

## 6-40.1 Object Model

The Object Model in Figure 6-40.1 represents a Fluid Flow Controller Device. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-40.1 Objects Present in a Fluid Flow Controller Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| Identity | Required | 1 |
| Message Router | Required | 1 |
| DeviceNet | Required | 1 |
| Connection | Required | at least<br>1 I/O and<br>1 Explicit |
| Assembly | Required | at least<br>1 Input and<br>1 Output |
| S-Device Supervisor | Required | 1 |
| S-Sensor Calibration *<br>or<br>S-Gas Calibration * | Optional | 0 or More |
| S-Analog Sensor | Required | 1 or More |
| S-Analog Actuator | Conditional ** | 1 |
| S-Single Stage Controller | Conditional ** | 1 |

* For FFC devices that flow **Liquid**, the **S-Sensor Calibration object** is used.
For FFC devices that flow **Gas**, the **S-Gas Calibration object** is used.

** Required for a Fluid Flow Controller, a device that contains a Valve and a Controller. Not supported in a Fluid Flow Meter Device.

## 6-40.2 Class Subclasses

Each class level subclass defines a unique meaning for an overlapping range of class attribute IDs and/or class service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given class is identified by the value of its Subclass class attribute. There are no class level subclasses specified for this device.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-40.3 Instance Subclasses

Each instance level subclass defines a unique meaning for an overlapping range of instance attribute IDs and/or instance service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given instance is identified by the value of its Subclass instance attribute.

The only instance level subclass specified for this device is:

- S-Analog Sensor Object, Instance Identifier 01 -- Subclass 01, Flow Diagnostics

### 6-40.4 Instance Identifiers

Additional objects are restricted for the purposes of reserving instance number ranges for future revisions of this device profile. As permitted by CIP, vendor-specific objects (object classes in the range 100-199 or 768-1279) may be added without instance number restrictions. However, open objects (i.e., all objects listed in Table 5A-1.1, but not included in Table 6-2.1, "Minimum Objects Required for All Devices") may only be added (as vendor-specific additions) using instance numbers in the range 100-199 or 768-1279, the vendor-specific open object instance number range for this device type. This allows object extensions to this device profile specification in future revisions (i.e., added open objects defined with instance identifiers < 100).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-40.1 Object Model for the FFC Device**

**Table 6-40.2 S-Analog Sensor Instances**

| S-Analog Sensor | Name | Description |
|---|---|---|
| Instance 1 * | Flow | Virtual sensor that calculates the fluid flow rate based on various sensed parameters, for example differential pressure |
| Instance 2 (optional) | Pressure | The measured fluid pressure |
| Instance 3 (optional) | Temperature | The measured fluid temperature |
| Instance 4 (optional) | Density | The measured fluid density |

* Subclass 01, Flow Diagnostics


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-40.5 How Objects Affect Behavior

**Table 6-40.3 Object Affect on Behavior**

| Object | Effect on behavior |
|---|---|
| Identity | Supports the Reset service. Upon receipt of a *Reset* Service Request of any *Type*, the Identity Object sends a *Reset* Service Request to the S-Device Supervisor. |
| Message Router | No effect |
| DeviceNet | Configures port attributes (node address, data rate, and BOI) |
| Connection Class | Contains the number of logical ports into or out of the device |
| Assembly | Defines input/output and configuration data format |
| S-Device Supervisor | Supports the Stop, Start, Reset, Abort, Recover and Perform_Diagnostic services for ALL Application Objects in the device and consolidates the Exception Conditions and Application Objects' Status.<br>This object behaves differently from the Identity Object in that the S-Device Supervisor object provides a single point of access to the Application Objects only; it does not effect the DeviceNet specific objects (i.e., Identity, DeviceNet, Connection, etc.). |
| S-Sensor Calibration | Modifies the correction algorithm of the related S-Analog Sensor object which includes the selection mechanism to enable an S-Sensor Calibration object instance. |
| S-Analog Sensor | A virtual sensor determines the measured fluid flow rate or other parameter. The flow sensor feeds the process variable to the S-Single Stage Controller object |
| S-Single Stage Controller | Feeds the control variable to the S-Analog Actuator object |
| S-Analog Actuator | Operates the Flow Control Valve of the device |

## 6-40.6 Defining Object Interfaces

**Table 6-40.4 Object Interfaces**

| Object | Interface |
|---|---|
| Identity | Message Router |
| Message Router | Explicit Messaging Connection Instance |
| DeviceNet | Message Router |
| Connection Class | Message Router |
| Assembly | I/O Connection or Message Router |
| S-Device Supervisor | Assembly or Message Router |
| S-Sensor Calibration | Message Router |
| S-Analog Sensor | Assembly or Message Router |
| S-Single Stage Controller | Assembly or Message Router |
| S-Analog Actuator | Assembly or Message Router |

## 6-40.7 I/O Assembly Instances

The following table identifies the I/O assembly instances supported by the FFC. As specified by the Assembly Object definition, the range of instances up to 99 are reserved for future specification revision to this profile.



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-40.5 FFC I/O Assembly Instances**

| Number | Required | Type | Name |
|---|---|---|---|
| 1 | Y | Input | Flow |
| 2 | Y (default) | Input | Status and Flow |
| 3 | N | Input | Status, Flow and Pressure |
| 4 | N | Input | Status, Flow and Temperature |
| 5 | N | Input | Status, Flow, Pressure and Temperature |
| 6 | N | Input | Status, Flow, Pressure and Valve |
| 7 | N | Input | Status, Flow, Pressure, Temperature and Valve |
| 8 | N | Input | Status, Flow, Setpoint and Valve |
| 9 | N | Input | Status, Flow, Setpoint, Override and Valve |
| 10 | C* (default) | Output | Setpoint |
| 11 | C* | Output | Override and Setpoint |
| 12 | N | Input | FP-Flow |
| 13 | N | Input | Status and FP-Flow |
| 14 | N | Input | Status, FP-Flow and FP-Pressure |
| 15 | N | Input | Status, FP-Flow and FP-Temperature |
| 16 | N | Input | Status, FP-Flow, FP-Pressure and FP-Temperature |
| 17 | N | Input | Status, FP-Flow, FP-Pressure and FP-Valve |
| 18 | N | Input | Status, FP-Flow, FP-Pressure, FP-Temperature and FP-Valve |
| 19 | N | Input | Status, FP-Flow, FP-Setpoint and FP-Valve |
| 20 | N | Input | Status, FP-Flow, FP-Setpoint, FP-Override and FP-Valve |
| 21 | N | Output | FP-Setpoint |
| 22 | N | Output | Override and FP-Setpoint |
| 23 | N | Input | Status, Flow, Density |
| 24 | N | Input | Status, Flow, Density, Temperature |
| 25 | N | Input | Status, Flow, Pressure, Temperature and Density |
| 26 | N | Input | Status, Flow, Pressure, Temperature, Density and Valve |
| 27 | N | Input | Status, FP-Flow, FP-Density |
| 28 | N | Input | Status, FP-Flow, FP-Density, FP-Temperature |
| 29 | N | Input | Status, FP-Flow, FP-Pressure, FP-Temperature and FP-Density |
| 30 | N | Input | Status, FP-Flow, FP-Pressure, FP-Temperature, FP-Density and FP-Valve |

* Required for a Fluid Flow Controller, a device that contains a Valve and a Controller. Not supported in a Fluid Flow Meter Device.

## 6-40.8 I/O Assembly Object Instance Data Attribute Encoding

The manufacturer of a Fluid Flow Controller Device must specify which Assembly instances are supported by the device.

The I/O Assembly DATA attribute is formatted as a structure of data elements with the first element ordered first. All data components are encoded with the Low-ordered Byte first; that is, the Least Significant Byte is in the Low-ordered Index (Address) i.e., Little Endian.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-40.9 Mapping I/O Assembly Data Attribute Components

Each of the *S-Analog Sensor*, *S-Analog Actuator* and *S-Single Stage Controller* object definitions specifies a behavior that modifies the *Data Type* of certain attributes based upon the first valid I/O connection established to an Assembly Object instance. In order to maintain consistency, this device type is limited to only allow connections to either INT or REAL based Assembly instances. Once a valid connection is established, attempts to configure connections to a different type of Assembly instance will return an error.

The following table indicates the I/O assembly Data attribute mapping for this FFC device.

**Table 6-40.6 I/O Assembly Data Mapping**

| Data Component | Class | | Instance | Attribute | | |
|---|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number | Type |
| Flow | S-Analog Sensor | $31_{hex}$ | 1 | Value | 6 | INT |
| Pressure | S-Analog Sensor | $31_{hex}$ | 2 | Value | 6 | INT |
| Temperature | S-Analog Sensor | $31_{hex}$ | 3 | Value | 6 | INT |
| Density | S-Analog Sensor | $31_{hex}$ | 4 | Value | 6 | INT |
| Valve | S-Analog Actuator | $32_{hex}$ | 1 | Value | 6 | INT |
| Override | S-Analog Actuator | $32_{hex}$ | 1 | Override | 5 | USINT |
| Setpoint | S-Single Stage Controller | $33_{hex}$ | 1 | Setpoint | 6 | INT |
| Status | S-Device Supervisor | $30_{hex}$ | 1 | Exception Status | 12 | BYTE |
| FP-Flow | S-Analog Sensor | $31_{hex}$ | 1 | Value | 6 | REAL |
| FP-Pressure | S-Analog Sensor | $31_{hex}$ | 2 | Value | 6 | REAL |
| FP-Temperature | S-Analog Sensor | $31_{hex}$ | 3 | Value | 6 | REAL |
| FP-Density | S-Analog Sensor | $31_{hex}$ | 4 | Value | 6 | REAL |
| FP-Valve | S-Analog Actuator | $32_{hex}$ | 1 | Value | 6 | REAL |
| FP-Setpoint | S-Single Stage Controller | $33_{hex}$ | 1 | Setpoint | 6 | REAL |

## 6-40.10 Object Limitations and Specific Definitions

This section describes limitations and specific definitions applicable to the listed objects of this device profile when these objects are applied in this type of device.

### 6-40.10.1 S-Device Supervisor Object Instance

**Table 6-40.7 Limitations**

| Attribute | Limitation |
|---|---|
| Device Type | Supported Values:<br>“FFC” = Fluid Flow Controller device (Liquid Type)<br>“FFM” = Fluid Flow Meter device (Liquid Type)<br>"FCG" = Fluid Flow Controller device (Gas Type)<br>"FMG" = Fluid Flow Meter device (Gas Type) |

### 6-40.10.2 Specific Definition

The following table specifies the data attribute bit mapping for the **Device Exception Detail** bytes for this FFC device. For more descriptive information, see the definition of the S-Device Supervisor Object Class. Noted, for each entry, is the Object from which the Status byte/bit is mapped. See the object specification for the detailed bit mapping.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Any Exception Bit not supported must default to 0. Note that this profile allows for only one byte of manufacturer specific exception detail.

**Table 6-40.8 Bit Mapping for Device Exception Detail**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| FFC Device Exception Detail Size | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| FFC Device Exception Detail [0] | Pressure High S-Analog Sensor 2 | Pressure Low S-Analog Sensor 2 | Valve High S-Analog Actuator | Valve Low S-Analog Actuator | Flow Control S-Single Stage Controller | Flow High S-Analog Sensor 1 | Flow Low S-Analog Sensor 1 | Reading Valid * S-Analog Sensor 1 |
| FFC Device Exception Detail [1] | 0 | 0 | 0 | 0 | Density High S-Analog Sensor 4 | Density Low S-Analog Sensor 4 | Temp High S-Analog Sensor 3 | Temp Low S-Analog Sensor 3 |
| FFC Device Exception Detail [2] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Manufacturer Exception Detail Size | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Manufacturer Exception Detail | 8 Bits defined by Manufacturer | | | | | | | |

* Only used in the Warning Exception Detail, this bit is always = 0 in the Alarm Exception Detail

### 6-40.10.3 S-Analog Sensor Object

**Table 6-40.9 Limitations Instance 1**

| Attribute | Limitation | Requirement | Default |
|---|---|---|---|
| Data Type | Supported Values = {INT; REAL} | Supported Values = {INT} | INT |
| Data Units | Supported Values = {Counts, % & Units of the Flow Group} (see Appendix D) | Supported Values = {counts} | Counts * |
| Subclass | | 01 = Flow Diagnostics | |

**Table 6-40.10 Limitations Instances 2**

| Attribute | Limitation | Requirement | Default |
|---|---|---|---|
| Data Type | Supported Values = {INT; REAL} | Supported Values = {INT} | INT |
| Data Units | Supported Values = {Counts, % & Units of the Pressure Group} (see Appendix D) | Supported Values = {counts} | Counts * |

**Table 6-40.11 Limitations Instance 3**

| Attribute | Limitation | Requirement | Default |
|---|---|---|---|
| Data Type | Supported Values = {INT; REAL} | Supported Values = {INT} | INT |
| Data Units | Supported Values = {Counts, % & Units of the Temperature Group} (see Appendix D) | Supported Values = {counts} | Counts * |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-40.12 Limitations Instance 4**

| Attribute | Limitation | Requirement | Default |
|---|---|---|---|
| Data Type | Supported Values = {INT; REAL} | Supported Values = {INT} | INT |
| Data Units | Supported Values = {Counts, % & Units of the Density Group} (see Appendix D) | Supported Values = {counts} | Counts * |

* Counts are interpreted as a "Percent of Full Scale" where a Count Value of 24576 (6000 hexadecimal) corresponds to 100%. This allows a maximum reading of 133.33%.

### 6-40.10.4 S-Analog Actuator Object

**Table 6-40.13 Limitations**

| Attribute | Limitation | Requirement | Default |
|---|---|---|---|
| Data Type | Supported Values = {INT; REAL} | Supported Values = {INT} | INT |
| Data Units | Supported Values = {Counts, %, Voltage, Current & Units of the Flow Group} (see Appendix D) | Supported Values = {counts} | Counts * |

* Counts are interpreted as a "Percent of Full Scale" where a Count Value of 24576 (6000 hexadecimal) corresponds to 100%. This allows a maximum reading of 133.33%.

### 6-40.10.5 S-Single Stage Controller Object

**Table 6-40.14 Limitations**

| Attribute | Limitation | Requirement | Default |
|---|---|---|---|
| Data Type | Supported Values = {INT; REAL} | Supported Values = {INT} | INT |
| Data Units | Supported Values = {Counts, %, & Units of the Flow Group} (see Appendix D) | Supported Values = {counts} | Counts * |
| Process Variable | Not accessible over the network. The *Process Variable* input to this object instance is the value of the S-Analog Sensor object instance *Value* attribute. | N.A. | N.A. |
| CV Data Type | Not supported | N.A. | N.A. |
| Control Variable | Not accessible over the network, The *Control Variable* output from this object is the value of the S-Analog Actuator object instance *Value* attribute. | N.A. | N.A. |

* Counts are interpreted as a "Percent of Full Scale" where a Count Value of 24576 (6000 hexadecimal) corresponds to 100%. This allows a maximum reading of 133.33%.

The **Data Type** and **Data Units** of the S-Single Stage Controller object shall remain consistent with those of the S-Analog Sensor object Instance 1 (Flow). Any change to the value of the Data Type or Data Units attribute of either of these two object instances will cause the device to self-modify the value of the corresponding attribute of the other object.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-40.11 Defining Device Configuration

Public access to the S-Device Supervisor, S-Analog Sensor, S-Analog Actuator, S-Single Stage Controller, and S-Sensor Calibration Objects by the Message Router must be supported for configuration of this device type.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-41 Enhanced Mass Flow Controller Device

**Device Type: 27 Hex**

## 6-41.1 Introduction

A Mass Flow Controller is a device that measures and controls the mass flow rate of gas or liquid. It contains three principle components: a mass flow rate sensor which can be one of a variety of types, including thermal or pressure-based; a mass flow rate metering valve which can be actuated by one of a variety of actuator types, including solenoid, voice coil or piezo; and, a controller which closes the loop by receiving a setpoint and driving the actuator such that the mass flow rate is controlled to the set point.

This Enhanced Mass Flow Controller Device Type differs from Mass Flow Controller Device Type $1A_{hex}$ by supporting both upstream process gas Temperature and upstream process gas Pressure in separate instances of the S-Analog Sensor Object.

## 6-41.2 Object Model

The Object Model in Figure 6-41.1 represents an Enhanced Mass Flow Controller Device. The table below indicates:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class

**Table 6-41.1 Objects Present in an Enhanced Mass Flow Controller Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| Identity | Required | 1 |
| Message Router | Required | 1 |
| DeviceNet | Required | 1 |
| Connection | Required | at least<br>1 I/O and<br>1 Explicit |
| Assembly | Required | at least<br>1 Input and<br>1 Output [1] |
| S-Device Supervisor | Required | 1 |
| S-Gas Calibration | Required | 1 or more |
| S-Analog Sensor | Required | 3<br>Instance 1 – flow<br>Instance 2 – pressure<br>Instance 3 - temperature |
| S-Analog Actuator | Conditional [1] | 1 |
| S-Single Stage Controller | Conditional [1] | 1 |

Required for a Mass Flow Controller, a device that contains a Valve and a Controller. Not supported in a Mass Flow Meter Device (an MFC without a Valve or a Controller).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-41.3 Class Subclasses

Each class level subclass defines a unique meaning for an overlapping range of class attribute IDs and/or class service IDs. The range for the subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given class is identified by the value of its Subclass class attribute. There are no class level subclasses specified for this device.

## 6-41.4 Instance Subclasses

Each instance level subclass defines a unique meaning for an overlapping range of instance attribute IDs and/or instance service IDs. The range for subclass definitions begins at ID 96 and numbers downward for attributes, and ID $63_{hex}$ and numbers downward for services. The subclass for a given instance is identified by the value of its Subclass instance attributes. The following tables identify which instance IDs are assigned subclasses for this device.

**Table 6-41.2 S-Analog Sensor Object Subclasses**

| Instance ID | Subclass Name | Subclass ID (Attribute 99 Value) | Required | Function | Restrictions |
|---|---|---|---|---|---|
| 1 | Flow Diagnostics | 01 | Required | Added diagnostics for EMFC | None |

**Table 6-41.3 S-Gas Calibration Object Subclass**

| Instance ID | Subclass Name | Subclass ID (Attribute 99 Value) | Required | Function | Restrictions |
|---|---|---|---|---|---|
| 1 | Standard T & P | 01 | Optional | Standard Temperature and Pressure | None |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-41.1 Object Model for the EMFC device**

## 6-41.5 How Objects Affect Behavior

**Table 6-41.4 Object Effect on Behavior**

| Object | Effect on behavior |
|---|---|
| Identity | Supports the Reset service. Upon receipt of a *Reset* Service Request of any *Type*, the Identity Object sends a *Reset* Service Request to the S-Device Supervisor. |
| Message Router | No effect |
| DeviceNet | Configures port attributes (node address, data rate, and BOI) |
| Connection Class | Contains the number of logical ports into or out of the device |
| | |
| Assembly | Defines input/output and configuration data format |
| S-Device Supervisor | Supports the Stop, Start, Reset, Abort, Recover and Perform_Diagnostic services for ALL Application Objects in the device and consolidates the Exception Conditions and Application Objects' Status.<br>This object behaves differently from the Identity Object in that the S-Device Supervisor object provides a single point of access to the Application Objects only; it does not affect the DeviceNet specific objects (i.e., Identity, DeviceNet, Connection, etc.). |
| S-Gas Calibration | Modifies the correction algorithm of the S-Analog Sensor object which includes the selection mechanism to enable an S-Gas Calibration object instance. |
| S-Analog Sensor | Feeds the process variable to the Single Stage Controller object |
| S-Single Stage Controller | Feeds the control variable to the Analog Actuator object |
| S-Analog Actuator | Operates the Flow Control Valve of the device |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 6-41.6 Defining Object Interfaces

**Table 6-41.5 Object Interfaces**

| Object | Interface |
|---|---|
| Identity | Message Router |
| Message Router | Explicit Messaging Connection Instance |
| DeviceNet | Message Router |
| Connection Class | Message Router |
| Assembly | I/O Connection or Message Router |
| S-Device Supervisor | Assembly or Message Router |
| S-Gas Calibration | Message Router |
| S-Analog Sensor | Assembly or Message Router |
| S-Single Stage Controller | Assembly or Message Router |
| S-Analog Actuator | Assembly or Message Router |

## 6-41.7 I/O Assembly Instances

The following table identifies the I/O assembly instances supported by the EMFC and EMFM devices.

**Table 6-41.6 I/O Assembly Instances**

| | Required | | | |
|---|---|---|---|---|
| Number | EMFC | EMFM | Type | Name |
| 1 | N | N | Input | Flow |
| 2 | Y (default) | Y (default) | Input | Status and Flow |
| 3 | N | N | Input | Status, Flow and Valve |
| 4 | N | N | Input | Status, Flow, and Setpoint |
| 5 | N | N | Input | Status, Flow, Setpoint and Valve |
| 6 | Y | N | Input | Status, Flow, Setpoint, Override and Valve |
| 7 | Y (default) | N | Output | Setpoint |
| 8 | Y | N | Output | Override and Setpoint |
| 9 | N | N | Input | Status |
| 10 | N | N | - | (assembly not used) |
| 11 | N | N | - | (assembly not used) |
| 12 | N | N | - | (assembly not used) |
| 13 | N | N | Input | FP-Flow |
| 14 | Y | Y | Input | FP-Status and Flow |
| 15 | N | N | Input | FP-Status, Flow and Valve |
| 16 | N | N | Input | FP-Status, Flow, and Setpoint |
| 17 | N | N | Input | FP-Status, Flow, Setpoint and Valve |
| 18 | Y | N | Input | FP-Status, Flow, Setpoint, Override and Valve |
| 19 | Y | N | Output | FP-Setpoint |
| 20 | Y | N | Output | FP-Override and Setpoint |
| 21 | Y | Y | Input | Status, Flow, Pressure and Temperature |
| 22 | N | N | Input | Status, Flow, Valve, Pressure and Temperature |
| 23 | Y | Y | Input | FP – Status, Flow, Pressure and Temperature |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Number | Required EMFC | Required EMFM | Type | Name |
|---|---|---|---|---|
| 24 | N | N | Input | FP – Status, Flow, Valve, Pressure and Temperature |
| 25 | N | N | Input | Exception Detail Alarm and Exception Detail Warning |
| Note that FP refers to “Floating Point”, which is analogous to the data type “REAL”. | | | | |

## 6-41.8 I/O Assembly Object Instance Data Attribute Format

The manufacturer of an Enhanced Mass Flow Controller Device must specify which Assembly instances are supported by the device.

The I/O Assembly Data attribute has the format shown below.

**Table 6-41.7 I/O Assembly Data Attribute**

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | Flow (low byte) | | | | | | | |
| | 1 | Flow (high byte) | | | | | | | |
| 2 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow (high byte) | | | | | | | |
| 3 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow (high byte) | | | | | | | |
| | 3 | Valve (low byte) | | | | | | | |
| | 4 | Valve (high byte) | | | | | | | |
| 4 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow (high byte) | | | | | | | |
| | 3 | Setpoint (low byte) | | | | | | | |
| | 4 | Setpoint (high byte) | | | | | | | |
| 5 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow (high byte) | | | | | | | |
| | 3 | Setpoint (low byte) | | | | | | | |
| | 4 | Setpoint (high byte) | | | | | | | |
| | 5 | Valve (low byte) | | | | | | | |
| | 6 | Valve (high byte) | | | | | | | |
| 6 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow (high byte) | | | | | | | |
| | 3 | Setpoint (low byte) | | | | | | | |
| | 4 | Setpoint (high byte) | | | | | | | |
| | 5 | Override | | | | | | | |
| | 6 | Valve (low byte) | | | | | | | |
| | 7 | Valve (high byte) | | | | | | | |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Enhanced Mass Flow Controller, Type: $27_{hex}$

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 0 | Setpoint (low byte) | | | | | | | |
| | 1 | Setpoint (high byte) | | | | | | | |
| 8 | 0 | Override | | | | | | | |
| | 1 | Setpoint (low byte) | | | | | | | |
| | 2 | Setpoint (high byte) | | | | | | | |
| 9 | 0 | Status | | | | | | | |
| 13 | 0 | Flow (low byte) | | | | | | | |
| | 1 | Flow | | | | | | | |
| | 2 | Flow | | | | | | | |
| | 3 | Flow (high byte) | | | | | | | |
| 14 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow | | | | | | | |
| | 3 | Flow | | | | | | | |
| | 4 | Flow (high byte) | | | | | | | |
| 15 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow | | | | | | | |
| | 3 | Flow | | | | | | | |
| | 4 | Flow (high byte) | | | | | | | |
| | 5 | Valve (low byte) | | | | | | | |
| | 6 | Valve | | | | | | | |
| | 7 | Valve | | | | | | | |
| | 8 | Valve (high byte) | | | | | | | |
| 16 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow | | | | | | | |
| | 3 | Flow | | | | | | | |
| | 4 | Flow (high byte) | | | | | | | |
| | 5 | Setpoint (low byte) | | | | | | | |
| | 6 | Setpoint | | | | | | | |
| | 7 | Setpoint | | | | | | | |
| | 8 | Setpoint (high byte) | | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Enhanced Mass Flow Controller, Type: $27_{hex}$

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 17 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow | | | | | | | |
| | 3 | Flow | | | | | | | |
| | 4 | Flow (high byte) | | | | | | | |
| | 5 | Setpoint (low byte) | | | | | | | |
| | 6 | Setpoint | | | | | | | |
| | 7 | Setpoint | | | | | | | |
| | 8 | Setpoint (high byte) | | | | | | | |
| | 9 | Valve (low byte) | | | | | | | |
| | 10 | Valve | | | | | | | |
| | 11 | Valve | | | | | | | |
| | 12 | Valve (high byte) | | | | | | | |
| 18 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow | | | | | | | |
| | 3 | Flow | | | | | | | |
| | 4 | Flow (high byte) | | | | | | | |
| | 5 | Setpoint (low byte) | | | | | | | |
| | 6 | Setpoint | | | | | | | |
| | 7 | Setpoint | | | | | | | |
| | 8 | Setpoint (high byte) | | | | | | | |
| | 9 | Override | | | | | | | |
| | 10 | Valve (low byte) | | | | | | | |
| | 11 | Valve | | | | | | | |
| | 12 | Valve | | | | | | | |
| | 13 | Valve (high byte) | | | | | | | |
| 19 | 0 | Setpoint (low byte) | | | | | | | |
| | 1 | Setpoint | | | | | | | |
| | 2 | Setpoint | | | | | | | |
| | 3 | Setpoint (high byte) | | | | | | | |
| 20 | 0 | Override | | | | | | | |
| | 1 | Setpoint (low byte) | | | | | | | |
| | 2 | Setpoint | | | | | | | |
| | 3 | Setpoint | | | | | | | |
| | 4 | Setpoint (high byte) | | | | | | | |
| 21 | 0 | status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow (high byte) | | | | | | | |
| | 3 | Pressure (low byte) | | | | | | | |
| | 4 | Pressure (high byte) | | | | | | | |
| | 5 | Temperature (low byte) | | | | | | | |
| | 6 | Temperature (high byte) | | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Enhanced Mass Flow Controller, Type: $27_{hex}$

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 22 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow (high byte) | | | | | | | |
| | 3 | Valve (low byte) | | | | | | | |
| | 4 | Valve (high byte) | | | | | | | |
| | 5 | Pressure (low byte) | | | | | | | |
| | 6 | Pressure (high byte) | | | | | | | |
| | 7 | Temperature (low byte) | | | | | | | |
| | 8 | Temperature (high byte) | | | | | | | |
| 23 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow | | | | | | | |
| | 3 | Flow | | | | | | | |
| | 4 | Flow (high byte) | | | | | | | |
| | 5 | Pressure (low byte) | | | | | | | |
| | 6 | Pressure | | | | | | | |
| | 7 | Pressure | | | | | | | |
| | 8 | Pressure (high byte) | | | | | | | |
| | 9 | Temperature (low byte) | | | | | | | |
| | 10 | Temperature | | | | | | | |
| | 11 | Temperature | | | | | | | |
| | 12 | Temperature (high byte) | | | | | | | |
| 24 | 0 | Status | | | | | | | |
| | 1 | Flow (low byte) | | | | | | | |
| | 2 | Flow | | | | | | | |
| | 3 | Flow | | | | | | | |
| | 4 | Flow (high byte) | | | | | | | |
| | 5 | Valve (low byte) | | | | | | | |
| | 6 | Valve | | | | | | | |
| | 7 | Valve | | | | | | | |
| | 8 | Valve (high byte) | | | | | | | |
| | 9 | Pressure (low byte) | | | | | | | |
| | 10 | Pressure | | | | | | | |
| | 11 | Pressure | | | | | | | |
| | 12 | Pressure (high value) | | | | | | | |
| | 13 | Temperature (low byte) | | | | | | | |
| | 14 | Temperature | | | | | | | |
| | 15 | Temperature | | | | | | | |
| | 16 | Temperature (high byte) | | | | | | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Instance | Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 25 | 0 | Status | | | | | | | |
| | 1 | Exception Detail Alarm 0 (size, common) | | | | | | | |
| | 2 | Exception Detail Alarm 1 (common 0) | | | | | | | |
| | 3 | Exception Detail Alarm 2 (common 1) | | | | | | | |
| | 4 | Exception Detail Alarm 3 (size, device) | | | | | | | |
| | 5 | Exception Detail Alarm 4 (device 0) | | | | | | | |
| | 6 | Exception detail Alarm 5 (device 1) | | | | | | | |
| | 7 | Exception Detail Alarm 6 (size, manufacturer) | | | | | | | |
| | 8 | Exception Detail Alarm 7 (manufacturer, 0) | | | | | | | |
| | 9 | Exception Detail Warning 0 (size, common) | | | | | | | |
| | 10 | Exception Detail Warning 1 (common 0) | | | | | | | |
| | 11 | Exception Detail Warning 2 (common 1) | | | | | | | |
| | 12 | Exception Detail Warning 3 (size, device) | | | | | | | |
| | 13 | Exception Detail Warning 4 (device 0) | | | | | | | |
| | 14 | Exception Detail Warning 5 (device 1) | | | | | | | |
| | 15 | Exception Detail Warning 6 (size, manufacturer) | | | | | | | |
| | 16 | Exception Detail Warning 7 (manufacturer, 0) | | | | | | | |

### 6-41.8.1 Mapping I/O Assembly Data Attribute Components

Each of the S-Analog Sensor, S-Analog Actuator and S-Single Stage Controller object definitions specifies a behavior that modifies the Data Type of certain attributes based upon the first valid I/O connection established to an Assembly Object instance. In order to maintain consistency, this device type will only allow connections to either INT or REAL based Assembly instances. Once a valid connection is established, attempts to configure connections to a different type of Assembly instance will return an error

The following table indicates the I/O assembly Data attribute mapping for this EMFC device.

**Table 6-41.8 I/O Assembly Data Mapping**

| Data Component | Class | | Instance | Attribute | | |
|---|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number | Type |
| Flow | S-Analog Sensor | $31_{hex}$ | 1 | Indicated Flow | 6 | INT |
| FP-Flow | S-Analog Sensor | $31_{hex}$ | 1 | Indicated Flow | 6 | REAL |
| Valve | S-Analog Actuator | $32_{hex}$ | 1 | Value | 6 | INT |
| FP-Valve | S-Analog Actuator | $32_{hex}$ | 1 | Value | 6 | REAL |
| Setpoint | S-Single Stage Controller | $33_{hex}$ | 1 | Setpoint | 6 | INT |
| FP-Setpoint | S-Single Stage Controller | $33_{hex}$ | 1 | Setpoint | 6 | REAL |
| Status | S-Device Supervisor | $30_{hex}$ | 1 | Exception Status | 12 | BYTE |
| Override | S-Analog Actuator | $32_{hex}$ | 1 | Override | 5 | USINT |
| Exception Detail Alarm | S-Device Supervisor | $30_{hex}$ | 1 | Exception Detail Alarm | 13 | STRUCT |
| Exception Detail Warning | S-Device Supervisor | $30_{hex}$ | 1 | Exception Detail Warning | 14 | STRUCT |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Data Component | Class | | Instance | Attribute | | |
|---|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number | Type |
| Pressure | S-Analog Sensor | $31_{hex}$ | 2 | Indicated Pressure | 6 | INT |
| FP–Pressure | S-Analog Sensor | $31_{hex}$ | 2 | Indicated Pressure | 6 | REAL |
| Temperature | S-Analog Sensor | $31_{hex}$ | 3 | Indicated Temperature | 6 | INT |
| FP-Temperature | S-Analog Sensor | $31_{hex}$ | 3 | Indicated Temperature | 6 | REAL |

## 6-41.9 Object Limitations and Specific Definitions

This section describes limitations and specific definitions applicable to the listed objects of this device profile when these objects are applied in this type of device.

### 6-41.9.1 S-Device Supervisor Object Instance

The following limitations apply:

**Table 6-41.9 Device Type Attribute Limitations**

| Attribute | Limitation |
|---|---|
| Device Type | Supported Values:<br>“EMFC” = Enhanced Mass Flow Controller device<br>“EMFM” = Enhanced Mass Flow Meter device |

**Specific Definitions**

The following table specifies the data attribute bit mapping for the Device Exception Detail bytes for this EMFC device. For more descriptive information, see the definition of the S-Device Supervisor Object Class. Noted, for each entry, is the Object from which the Status byte/bit is mapped. See the object specification for the detailed bit mapping.

Any Exception Bit not supported must default to 0. Note that this profile allows for only one byte of manufacturer specific exception detail.

Note that the status bits for the Pressure (Instance 2 of the S-Analog Sensor object) and the Temperature (Instance 3 of the S-Analog Sensor object) shall use the same bit mapping as the Flow instance (Instance 1 of the S-Analog Sensor object).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 6-41.10 Device Exception Detail Mapping**

| Data Component | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| MFC Device Exception Detail Size | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| MFC Device Exception Detail Byte 0 | Reserved 0 | Reserved 0 | Valve High S-Analog Actuator | Valve Low S-Analog Actuator | Flow Control S-Single Stage Controller | Flow High S-Analog Sensor | Flow Low S-Analog Sensor | Not Reading Valid [1] S-Analog Sensor Flow |
| MFC Device Exception Detail Byte 1 | Reserved 0 | Reserved 0 | Not Reading Valid S-Analog Sensor temperature | Not Reading Valid S-Analog Sensor pressure | Gas Temp High S-Analog Sensor Instance 3 | Gas Temp Low S-Analog Sensor Instance 3 | Pressure High S-Analog Sensor Instance 2 | Pressure Low S-Analog Sensor Instance 2 |
| Manufacturer Exception Detail Size | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Manufacturer Exception Detail | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table Footnotes

1. Only used in the Warning Exception Detail, this bit is always = 0 in the Alarm Exception Detail. Refer to the S-Analog Sensor instance.

### 6-41.9.2 S-Analog Sensor Object

The following limitations apply:

**Table 6-41.11 S-Analog Sensor Object Attribute Limitations for Instance 1, (flow)**

| Attribute | Limitation | Requirement | Default |
|---|---|---|---|
| Data Type | Supported Values = {INT; REAL} | Supported Values = {INT; REAL} | INT |
| Data Units | Supported Values = {Counts & Units of the Flow Group} (see Appendix D) | Supported Values = {Counts; sccm} | Counts |
| Offset-A Data Type | Supported Values = {INT; REAL} | Supported Values = {INT; REAL} | INT |
| Gain Data Type | Supported Values = {INT; REAL} | Supported Values = {INT; REAL} | INT |



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-41.9.3 S-Analog Sensor Object

The following limitations apply:

**Table 6-41.12 S-Analog Sensor Object Attribute Limitations for Instance 2, (pressure)**

| Attribute | Limitation | Requirement | Default |
|---|---|---|---|
| Data Type | Supported Values = {INT; REAL} | Supported Values = {INT; REAL} | INT |
| Data Units | Supported Values = {Counts & Units of the Pressure Group} (see Appendix D) | Supported Values = {Counts; psi} | Counts |
| Offset-A Data Type | Supported Values = {INT; REAL} | Supported Values = {INT; REAL} | INT |
| Gain Data Type | Supported Values = {INT; REAL} | Supported Values = {INT; REAL} | INT |

### 6-41.9.4 S-Analog Actuator Object

The following limitations apply:

**Table 6-41.13 S-Analog Actuator Object Attribute Limitations**

| Attribute | Limitation | Requirement | Default |
|---|---|---|---|
| Data Type | Supported Values = {INT; REAL} | Supported Values = {INT; REAL} | INT |
| Data Units | Supported Values = {Counts, %, Voltage, Current & Units of the Flow Group} (see Appendix D) | Supported Values = {Counts; %} | Counts |
| Gain Data Type | Supported Values = {INT; REAL} | Supported Values = {INT; REAL} | INT |

### 6-41.9.5 S-Single Stage Controller Object

The following limitations apply:

**Table 6-41.14 S-Single Stage Controller Object Attribute Limitations**

| Attribute | Limitation | Requirement | Default |
|---|---|---|---|
| Data Type | Supported Values = {INT; REAL} | Supported Values = {INT; REAL} | INT |
| Data Units | Supported Values = {Counts, %, Voltage, Current & Units of the Flow Group} (see Appendix D) | Supported Values = {Counts; %} | Counts |
| Process Variable | Implementation optional. The *Process Variable* input to this object instance is the value of the S-Analog Sensor object instance *Value* attribute. | N.A. | N.A. |
| CV Data Type | Not supported | N.A. | N.A. |
| Control Variable | Not accessible over the network. The *Control Variable* output from this object is the value of the S-Analog Actuator object instance *Value* Attribute. | N.A. | N.A. |

Note - % refers to a vendor specific value that equates to 0 to 100% of full scale

## 6-41.10 Defining Device Configuration

Public access to the S-Device Supervisor, S-Analog Sensor, S-Analog Actuator, S-Single Stage Controller, and S-Gas Calibration Objects by the Message Router must be supported for configuration of this device type.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-42 CIP-Modbus Translator Device

**Device Type: 29 Hex**

This profile identifies the requirements for a standalone CIP-Modbus translator. The CIP-Modbus translator routes CIP requests to a Modbus network so the Modbus devices appear as CIP devices to CIP Originators. Other device types may also include this functionality.

## 6-42.1 Object Model

The Object Model in Figure 6-42.1 represents the minimum support in a CIP-Modbus Translator Device. Products that report other device types may also support CIP Modbus translation functionality. The purpose of this new device type is for devices whose primary application is CIP Modbus translation. If other devices choose to include the CIP-Modbus translator functionality the requirements are located here. The table and the following paragraph below indicate:

- the object classes present in this device
- whether or not the class is required
- the number of instances present in each class
- the CIP-Modbus Translation Function defined in Volume 7

**Table 6-42.1 Objects Present in a CIP Modbus Translator Device**

| Object Class | Optional/Required | # of Instances |
|---|---|---|
| CIP Required Objects | Required | See Section 6-2.1 |
| Port Object | Required | at least 2 [1] |
| Modbus Serial Link Object | Optional | at least 1 |

Table Footnotes
1 At least one must be a Modbus type (Serial or TCP) and one must be a CIP type (e.g. EtherNet/IP)

**Important:** The CIP_Modbus Translation Function is required. The translation function shall comply with all the requirements of Volume 7, Integration of Modbus Devices into the CIP Architecture.

The CIP-Modbus Translator profile does not specify any instances of the Assembly Object or any application objects outside those listed above. Its sole purpose is to route CIP requests to a Modbus network so that the Modbus devices appear as CIP devices to a CIP Originator.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 6-42.1 Object Model for CIP-Modbus Translator Device**

## 6-42.2 How Objects Affect Behavior

The objects for this device affect the device's behavior as shown in the table below.

**Table 6-42.2 Object Affect on Behavior**

| Object | Effect on Behavior |
|---|---|
| CIP Required Objects | See Section 6-2.2 for behavior |
| Port Object | Determines the port used for translation path |
| Modbus Serial Link Object | Configures parameters for Modbus Serial Link master functionality |

## 6-42.3 Defining Object Interfaces

The objects in the CIP-Modbus translator device have the interfaces listed in the following table.

**Table 6-42.3 Object Interfaces**

| Object | Interface |
|---|---|
| CIP Required Objects | See Section 6-2.3 for details |
| Port Object | Message Router |
| Modbus Serial Link Object | Message Router |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 6-43 Embedded Component

## Device Type: C8 Hex

This profile is used to identify embedded components within a device, hence it is not a full device profile. An embedded component may be hardware or software. This profile simply defines the changes to the requirements of the Identity object attributes and services for the purposes of reporting the revision of an embedded component via an additional Identity object (see section 5A-2) instance.

Where the requirements are not specifically stated within the sub-sections that follow, the requirements for instance one (1) of the Identity object apply. For example, the Access Rule and NV requirements for the Instance Attributes are the same except where indicated otherwise.

## 6-43.1 Identity Object Instance Attribute Requirements

Requests to optional attributes that are not supported by a particular Embedded Component instance shall return Attribute Not Supported (General Status code 0x14).

**Table 6-43.1 Identity Object Instance Attribute Requirements**

| Attr ID | Name | Requirements |
|---|---|---|
| 1 | Vendor ID | Required. The Vendor ID of the device (Instance 1) shall be used. |
| 2 | Device Type | Required. Shall be 0xC8. |
| 3 | Product Code | Required. Product Code of the embedded component, see section 6-43.1.1. |
| 4 | Revision | Required. Revision of the embedded component, see section 6-43.1.2. |
| 5 | Status | Required. If the embedded component does not support reporting Status per section 5A-2.2.1.5, then the value 0x0080 shall be returned. The value 8 in the Extended Device Status field indicates that the Status attribute is Not Applicable. |
| 6 | Serial Number | Required. May be 0 if the embedded component does not have a serial number or its serial number cannot be reported as a UDINT. |
| 7 | Product Name | Required. This text string should represent a short description of the embedded component represented by the Product Code in attribute 3. |
| 8 | State | Optional. If implemented, the State of the embedded component. |
| 9 | Configuration Consistency Value | Optional. If implemented, the Configuration Consistency Value of the embedded component. |
| 10 | Heartbeat Interval | Optional. If implemented, the Heartbeat Interval of the embedded component. |
| 11 | Active Language | Optional. If implemented, the value reflects the language for only the Product Name attribute of this instance. Set is optional. |
| 12 | Supported Language List | Optional. If implemented, the values reflect the languages that are supported for the string attributes of this instance only. |
| 13 | International Product Name | Optional. If implemented, the names of the embedded component in various languages. |
| 14 | Semaphore | Optional. If implemented, the Semaphore of the embedded component. |
| 15 | Assigned_Name | Optional. If implemented, applies to only the embedded component. |
| 16 | Assigned_Description | Optional. If implemented, applies to only the embedded component. |
| 17 | Geographic_Location | Not allowed. Return an error response specifying Attribute Not Supported (General Status code 0x14). |
| 18 | Modbus Id Info | Not allowed. Return an error response specifying Attribute Not Supported (General Status code 0x14). |
| 19 | Protection Mode | Not allowed. Return an error response specifying Attribute Not Supported (General Status code 0x14). |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 6-43.1.1 Product Code (Attribute 3)

The Product Code of an embedded component is assigned by the device manufacturer, i.e. the vendor incorporating the embedded component into the device. Product Codes are unique within the Embedded Component Device Type (0xC8) and the Vendor ID. For a given embedded component, the device manufacturer may choose to share the same Product Code across multiple products or assign the Product Code on a per product basis.

### 6-43.1.2 Revision (Attribute 4)

When the native revision format of the embedded component is the same as the Revision attribute format, the Revision attribute shall reflect the native revision value of the embedded component. Zero (0) is allowed for the Minor Revision value of embedded component instances.

When the native revision format of the embedded component is not the same as the Revision attribute format, it is recommended that the native revision value of the embedded component be mapped to the Revision attribute format. For example, if the native revision format values follow the "A, B, C…" pattern, these could be mapped to Revision values "1.000, 2.000, 3.000…"

When it is not possible to provide a mapping from the native revision format of an embedded component to the Revision attribute format, both the Major and Minor Revision values shall be set to zero (0) and the native revision value shall be included in the Product Name attribute.

## 6-43.2 Identity Object Instance Service Requirements

The service requirements for embedded component Identity object instances are the same as for instance one (1), except for the Reset service as specified in Table 6-43.2.

**Table 6-43.2 Identity Object Reset Service Requirements**

| Service Code | Need in Implementation – Instance | Service Name | Description of Service |
|---|---|---|---|
| 0x05 | Optional | Reset | If supported, the reset applies to the scope of the embedded component. Return "Service not Supported for Specified Path" (0x2E) if the embedded component instance does not support the Reset service. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This page is intentionally left blank


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Volume 1: Common Industrial Protocol (CIP™) Specification

# Chapter 7: Electronic Data Sheets


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Contents


7-1 Introduction ..... 5
7-1.1 Terms and Definitions Used in this Chapter ..... 5
7-2 Configuration Options ..... 6
7-2.1 Configuration Data Sources and Methods ..... 6
7-2.1.1 Configuration Support Using Printed Data Sheets ..... 6
7-2.1.2 Configuration Support Using an Electronic Data Sheet ..... 7
7-2.1.3 Configuration Support Using Parameter Objects and Parameter Object Stubs ..... 8
7-2.1.3.1 Using Full Parameter Objects ..... 8
7-2.1.3.2 Using Parameter Object Stubs ..... 9
7-2.1.4 Configuration Using an EDS and Parameter Object Stubs ..... 9
7-2.1.5 Configuration Using a Configuration Assembly ..... 10
7-2.1.6 Configuration Using a Device Type Manager (DTM) ..... 11
7-2.1.6.1 Developing a Device Type Manager (DTM) ..... 11
7-3 Electronic Data Sheet Definition ..... 12
7-3.1 The Electronic Data Sheet ..... 12
7-3.2 The Product Data Sheet Model ..... 12
7-3.3 Using an EDS with Configuration Tools ..... 15
7-3.3.1 EDS Interpreter Functions ..... 15
7-3.3.2 EDS File Organization ..... 16
7-3.3.2.1 EDS Content ..... 16
7-3.4 File Naming Requirements ..... 19
7-3.5 EDS Data Encoding Requirements ..... 19
7-3.5.1 Elementary Data Type Encodings ..... 19
7-3.5.1.1 Character String Conventions ..... 19
7-3.5.1.2 BOOL, USINT, UINT, UDINT, ULINT – Unsigned Integer Value Convention ..... 22
7-3.5.1.3 SINT, INT, DINT, LINT – Signed Integer Value Convention ..... 22
7-3.5.1.4 BYTE, WORD, DWORD, LWORD – Bit String Convention ..... 23
7-3.5.1.5 REAL, LREAL – Floating Point Value Convention ..... 23
7-3.5.1.6 STIME – Synchronous Time convention ..... 24
7-3.5.1.7 DATE – Date convention ..... 25
7-3.5.1.8 TIME_OF_DAY – Time of Day convention ..... 25
7-3.5.1.9 DATE_AND_TIME – Date and Time of Day convention ..... 25
7-3.5.1.10 FTIME – Duration (high resolution) convention ..... 25
7-3.5.1.11 LTIME – Duration (long) convention ..... 26
7-3.5.1.12 ITIME – Duration (short) convention ..... 26
7-3.5.1.13 TIME – Duration (milliseconds) convention ..... 26
7-3.5.1.14 EPATH – Path Format Convention ..... 27
7-3.5.1.15 Array of EPATHs – Array of Paths Format Convention ..... 27
7-3.5.1.16 ENGUNIT – Engineering Units convention ..... 28
7-3.5.1.17 EDS_External_Id – Non-CIP Modular Module Identifier ..... 28
7-3.5.2 Defined Constructed Data Type Encodings ..... 28
7-3.5.2.1 SIGNED_ODOMETER – Signed large magnitude counters ..... 28
7-3.5.2.2 ODOMETER – Large magnitude counters ..... 28
7-3.5.3 EDS-specific Data Type Encodings ..... 29
7-3.5.3.1 EDS_Date Convention ..... 29
7-3.5.3.2 EDS_Time_Of_Day Convention ..... 29
7-3.5.3.3 EDS_URL – Uniform Resource Locator ..... 29
7-3.5.3.4 EDS_Revision – Revision ..... 29
7-3.5.3.5 EDS_Field_Keyword - Field Keyword Convention ..... 30
7-3.5.3.6 EDS_Service – Service Keyword Convention ..... 30
7-3.5.3.7 EDS Entry Keyword Reference Convention ..... 30
7-3.5.4 EDS Comments ..... 31
7-3.6 Basic EDS File Requirements ..... 31



SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

7-3.6.1 Common Object Class Information ..... 31
7-3.6.1.1 Public Object Class Sections ..... 31
7-3.6.1.2 Vendor Specific Object Class Sections ..... 35
7-3.6.1.3 Common Object Class Entry Keywords ..... 36
7-3.6.2 File Description Section ..... 46
7-3.6.3 Device Description Section ..... 49
7-3.6.4 Device Classification Section ..... 53
7-3.6.4.1 Classification Field ..... 53
7-3.6.5 Parameter Class Section ..... 53
7-3.6.5.1 Descriptor Entry Keyword ..... 54
7-3.6.5.2 CfgAssembly Entry Keyword ..... 54
7-3.6.5.3 SafetyCfgAssembly Entry Keyword ..... 54
7-3.6.5.4 Configuration Assembly Extension A ..... 54
7-3.6.6 Parameters Section ..... 56
7-3.6.6.1 Param Keyword ..... 57
7-3.6.6.2 Enum Keyword ..... 59
7-3.6.6.3 Fixed Keyword ..... 60
7-3.6.6.4 ConstructedParam Keyword ..... 61
7-3.6.7 Parameter Groups Section ..... 65
7-3.6.7.1 Group Type Entry Keywords ..... 65
7-3.6.8 Assembly Section ..... 66
7-3.6.8.1 Assem type Entry Keywords ..... 67
7-3.6.8.2 Variant, VariantExa and BitStringVariant Entry Keywords ..... 73
7-3.6.8.3 Array Entry Keyword ..... 83
7-3.6.8.4 Examples of Adapter/Rack Assem Data Formatting ..... 87
7-3.6.8.5 Example of Indirect Parameter Handling ..... 91
7-3.6.9 Complete Electronic Data Sheet Example ..... 93
7-3.6.10 Connection Manager Section ..... 94
7-3.6.10.1 Connection Entry Keywords ..... 95
7-3.6.10.2 Production Inhibit Time in Milliseconds Network Segment Entry Keyword ..... 102
7-3.6.10.3 Production Inhibit Time in Microseconds Network Segment Entry Keyword ..... 103
7-3.6.10.4 Max Connection Establishment Time Entry Keyword ..... 103
7-3.6.11 Port Specific Connection Manager Section ..... 103
7-3.6.12 Port Section ..... 105
7-3.6.13 Capacity Section ..... 107
7-3.6.13.1 Traffic Spec ..... 107
7-3.6.13.2 Generating TSpec Values ..... 108
7-3.6.13.3 Connection Overhead ..... 109
7-3.6.13.4 Determining ConnOverhead Values ..... 110
7-3.6.13.5 Capacity Assumptions ..... 110
7-3.6.13.6 Calculating Implicit Communications Usage ..... 111
7-3.6.13.7 Connection Capacity Keywords ..... 111
7-3.6.13.8 Considerations for Multiport Devices ..... 113
7-3.6.14 Connection Configuration Class Section ..... 113
7-3.6.14.1 Mapping Attribute Format Number Entry Keyword ..... 114
7-3.6.14.2 Originator to Target Image Sizes and Paths Entry Keyword ..... 114
7-3.6.14.3 Target to Originator Image Sizes and Paths Entry Keyword ..... 115
7-3.6.14.4 Connection Target Name Entry Keyword ..... 115
7-3.6.14.5 Redundant Owner Connection Support Entry Keyword ..... 116
7-3.6.15 Event Enumeration Section ..... 117
7-3.6.16 Symbolic Translation Section ..... 120
7-3.6.17 Internationalization Section ..... 122
7-3.6.17.1 Descriptive Text Entry Keyword ..... 123
7-3.6.17.2 Device Type String Entry Keyword ..... 124
7-3.6.17.3 Product Name Entry Keyword ..... 124
7-3.6.17.4 Parameter Entry Keywords ..... 124


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

7-3.6.17.5 Constructed Parameter Entry Keywords ..... 125
7-3.6.17.6 Enumeration Entry Keyword ..... 126
7-3.6.17.7 Group Entry Keyword ..... 126
7-3.6.17.8 Assembly Entry Keywords ..... 126
7-3.6.17.9 Variant Entry Keywords ..... 127
7-3.6.17.10 Array Entry Keywords ..... 127
7-3.6.17.11 Connection Entry Keyword ..... 127
7-3.6.17.12 Port Entry Keyword ..... 128
7-3.7 Modular EDS File Requirements ..... 128
7-3.7.1 Modular Section ..... 128
7-3.7.2 Modular Additions to Basic EDS Sections ..... 130
7-3.7.2.1 Path format convention addition for Modular EDS Files – (EPATH) ..... 130
7-3.7.2.2 Additions to the Modular Section ..... 131
7-3.7.2.3 Additions to the Parameter Section ..... 132
7-3.7.2.4 Additions to the Assembly Section ..... 134
7-3.7.2.5 Additions to the Connection Manager Section ..... 137


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 7-1 Introduction

This chapter describes:

- standardized methods for device configuration
- the structure of an Electronic Data Sheet (EDS)

The information provides reference material for:

- product developers
- configuration tool designers

## 7-1.1 Terms and Definitions Used in this Chapter

**Table 7-1.1 Terms and Definitions**

| Term | Meaning |
|---|---|
| decoded format | The decoded attribute data values in CIP message format |
| EDS | The abbreviation for Electronic Data Sheet, a file on disk that contains configuration data for specific device types |
| encoded | The encoded attribute data values in the Electronic Data Sheet Format |
| full parameter object | A parameter object instance within a device that contains the data value, prompt strings, data conversion factors, and other information associated with the parameter |
| stub parameter object | A shorthand form of a parameter object instance that stores only fields 1-6 |
| Rack | Typically a multi-slot chassis that contains a variety of modules |
| Adapter | A device connected directly to a CIP network, which can be placed in one or more slots of a rack |
| Module | A device that can be placed in one or more slots of a rack |
| Adapter Rack Connection | A connection to an adapter whose O→T and T→O application data includes data produced/consumed by one or more modules |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 7-2 Configuration Options

The CIP standard allows for remote device configuration across the network and for embedding configuration parameters in devices. Using these features, you can select and modify device configuration settings for use in a particular application. The CIP interface allows access to a device's configuration settings.

**Figure 7-2.1 Illustration of Remote Device Configuration**

Devices containing configuration settings accessible only through the CIP communications interface require a configuration tool to change these settings. Devices configured only through the use of external switches, jumper blocks, thumbwheels, or other proprietary interfaces do not require a configuration client (tool) to modify the device's configuration settings. However, the device designer must provide a means for a tool to access and determine the state of hardware configuration switches.

### 7-2.1 Configuration Data Sources and Methods

This section describes possible methods for storing and accessing a device's configuration data

- A printed data sheet
- An Electronic Data Sheet (EDS)
- Parameter Objects and Parameter Object Stubs
- A combination of an EDS and Parameter Object Stubs
- A Configuration Assembly and any of the above methods
- A Device Type Manager (DTM) loaded into a Field Device Tool software package

#### 7-2.1.1 Configuration Support Using Printed Data Sheets

When using configuration information collected on a printed data sheet, configuration tools can only provide prompts for service, class, instance, and attribute data and relay this data to a device. A configuration tool of this type does not determine the context, content, or format of the data. The figure below shows the relationship between configuration information and the tool.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-2.2 Illustration of Printed Data Sheet Support**

### 7-2.1.2 Configuration Support Using an Electronic Data Sheet

You can provide configuration support for your device by using a specially formatted ASCII file, referred to as the *Electronic Data Sheet* (EDS). An EDS provides information about the device configuration data's:

- context
- content
- format

The information in an EDS allows configuration tools to provide informative screens that guide a user through the steps necessary to configure a device. Section 7-3 describes electronic data sheets.

An EDS provides all of the information necessary to access and alter the configurable parameters of a device. This information matches the information provided by instances of the Parameter Object Class. The CIP Object Library describes the Parameter Object Class in detail.

Manufacturers who choose not to supply an EDS on computer-readable media can provide a printed listing of their EDS so that the end user may create the computer-readable EDS using a text editor.

The following figure shows device configuration by a configuration tool that supports an EDS. The application objects in the device represent the destination addresses for configuration data. These addresses are encoded in the EDS.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-2.3 Illustration of Using Electronic Data Sheets for Configuration**

### 7-2.1.3 Configuration Support Using Parameter Objects and Parameter Object Stubs

The device's public *Parameter Object*, which is an optional data structure in a device, provides a third method for accessing the configuration data of a device. When a device uses parameter objects, it requires one instance of the Parameter Object class for each supported configuration parameter. Each instance is linked to a configurable parameter that can be an attribute of one of the device's other objects. Changing the Parameter Value attribute of a Parameter Object causes a corresponding change in the attribute value, as indicated by the Link Path attribute.

A *Full Parameter Object* contains all of the information necessary for device configuration. The CIP Object Library, contains a complete definition of the Parameter Object. A partially defined Parameter Object, called a *Parameter Object Stub*, contains the portion of configuration information needed to perform configuration that excludes user prompts, limit testing, and descriptive text that guides a user through the configuration.

#### 7-2.1.3.1 Using Full Parameter Objects

Parameter Objects embed all of the needed configuration information within the device. Parameter Objects provide:

- a known public interface to a device's configuration data values
- descriptive text
- data limits, default, minimum, and maximum values

**Important:** When a device contains full Parameter Objects, a configuration tool may derive all of the needed configuration information directly from the device.

The following figure shows the configuration of a device by a configuration tool that supports full Parameter Objects.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-2.4 Illustration of Using Parameter Objects To Configure a Device**

### 7-2.1.3.2 Using Parameter Object Stubs

Parameter Object stubs provide an established address to a device's configuration data values without requiring specification of descriptive text, data limits, and other parameter properties. When a device contains Parameter Object stubs, a configuration tool may obtain additional configuration information from an EDS or provide only a minimal interface to a parameter for modification.

### 7-2.1.4 Configuration Using an EDS and Parameter Object Stubs

A configuration tool may obtain information from partial Parameter Objects or Parameter Object stubs embedded in a device that provides a companion EDS. The EDS supplies additional parameter information needed by a configuration tool. The Parameter Object stubs can provide a known public interface to a device's parameter data values, while the EDS supplies descriptive text, data limits, and other parameter properties such as the:

- Data type and size for data validation
- Default data selections
- Descriptive user prompts
- Descriptive help text
- Descriptive parameter names

The figure below shows the configuration of a device by a configuration tool that supports Parameter Object stubs with electronic data sheets.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-2.5 Illustration of EDS and Parameter Object Stubs to Configure a Device**

### 7-2.1.5 Configuration Using a Configuration Assembly

An instance of the Configuration Assembly allows bulk upload and download of configuration data. If you use this method for device configuration, you shall document:

- The format of the configuration data blocks
- The address mapping for each configurable attribute

When specifying the data attribute of the Configuration Assembly, you shall:

- List the data components in the order given by the attribute block
- List data components larger than one byte starting with the low order byte
- List data components smaller than one byte justified to the right within a byte, starting with bit 0

Show the format of the data blocks in a table like the following:

**Table 7-2.1 Example of Configuration Assembly Data Attribute**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Operate Mode |
| | | | | o<br>o<br>o | | | | |
| n | Input Range | | | | | | | |

In addition to specifying the device's Configuration Assembly's data format, you shall map the individual configuration data components to their associated objects. Do this by specifying the Class, Instance, and Attribute IDs for each data component. This is essentially the same as specifying the Member_List attribute of the Assembly Object.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The table below shows an example mapping of Configuration Assembly data attribute components:

**Table 7-2.2 Example of Configuration Assembly Data Attribute Components**

| Data Component | Class | | Instance | Attribute | | Data |
|---|---|---|---|---|---|---|
| Name | Name | Number | Number | Name | Number | Type |
| Output Range | Analog Output | 0B hex | 1 | Output Range | 7 | UINT |
| | | o<br>o<br>o | | | | |
| Input Range | Analog Input | 0A hex | 1 | Input Range | 7 | BYTE |

### 7-2.1.6 Configuration Using a Device Type Manager (DTM)

A device can be configured by use of a Device Type Manager (DTM) software component. The DTM is loaded into a configuration software package that supports the Field Device Tool (FDT) interfaces to the DTM. This is demonstrated in below.

**Figure 7-2.6 Configuration Using a DTM Software Component**

Like an EDS file, the DTM contains all of the device specific information for configuring the device. However, while the EDS file is a structured text file describing the data objects, the DTM is a software component that adheres to the FDT interfaces. This allows a greater level of flexibility to add device specific functionality into the configuration interface then can be achieved in an EDS file.

#### 7-2.1.6.1 Developing a Device Type Manager (DTM)

The controlling specifications for the development of a DTM are in the FDT specification and the supporting annex for CIP networks. See www.fdtgroup.org for details.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 7-3 Electronic Data Sheet Definition

This section contains the formal specification of the electronic data sheet (EDS). The EDS allows a configuration tool to automate the device configuration process. The EDS specification provides an open standard for device configuration and compatibility among all CIP products.

## 7-3.1 The Electronic Data Sheet

An EDS may contain vendor-specific information in addition to the required device parameter information defined by this specification. The figure below shows a general block diagram of a sample EDS.

**Figure 7-3.1 General Block Diagram of an EDS File**

## 7-3.2 The Product Data Sheet Model

Electronic Data Sheets follow a product data sheet metaphor, modified to accommodate CIP requirements. Typically, a product data sheet provides a user with information needed to determine the product's features and the range of user-assigned values that may be selected for these features.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-3.2 Example of a Product Data Sheet**

| | | |
|---|---|---|
| Product Name: | Smart Widget I1 Series Z99 Dual Mode | <= Name of the product |
| Vendor: | Always Profitable Widget Works, Inc. | <= Who made it |
| Model: | Z99 DM | <= Catalog number |
| Revision: | 2.1 | <= Revision level |
| Serial Number: | 00123456789A | <= Unique serial number |

| User Settings | Minimum | Maximum | Factory Default | Access Point |
|---|---|---|---|---|
| 1. Gain | 1 | 64 | 32 | SW1-8 |
| 2. Scale Factor | 1 | 8 | 1 | SW4 |
| 3. Output Mode | RTZ | NRTZ | RTZ | Jumper Block 5 |
| 4. Output Frequency | 1 Hz | 100 Hz | 30 Hz | SW2 |
| 5. Status Register | N/A | N/A | N/A | LED2 |

The data sheet conveys information from the product manufacturer to the product user. The product user interprets the manufacturer's data sheet, decides which settings must be set to non-default values, and performs the necessary actions to get the information from the data sheet into the device.

A configuration tool that uses an EDS and parameter object stubs follows this same sequence, using an electronic form of the data sheet.

To perform the actual configuration, a configuration tool uses CIP messaging to perform the changes in the device. Currently, the textual information in an EDS must be ASCII-representable characters. Future revisions of this specification will support non-ASCII character representations such as UNICODE.

The EDS serves two main purposes by providing information needed to:

- Describe each device parameter, including its legal and default values
- Provide the user with a selection of choices for each configurable parameter in a device

The figure below compares the interpretation of a printed data sheet by a human with the interpretation of an EDS by a configuration tool. At the minimum, a CIP configuration tool must:

- Load the EDS into the configuration tool's memory
- Interpret the EDS contents to determine the characteristics of each parameter
- Present to the user a list of choices or data entry fields for each device parameter
- Load the user's parameter choices to the correct parameter addresses in the device


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-3.3 Illustration of Using Printed Data Sheets vs EDS**

Product developers may add optional EDS-related functions to a configuration tool. The specified EDS requirements provide a consistent and compatible approach for performing configuration in the CIP environment. This specification enforces only the requirements critical to achieving:

- Consistency among vendors for the benefit of device makers
- Compatibility among vendors for the benefit of tool makers

All EDS developers shall implement EDSs compliant with these requirements. Product developers determine all other implementation details. Every EDS interpreter designed for CIP products shall be capable of:

- Reading and interpreting any standard EDS
- Presenting information and choices to device users
- Building the messages necessary to configure the associated CIP product

The specification, in CIP Communication Model and Protocol, imposes strict requirements and protocols for transmitting messages over the CIP network. The figure below shows the compliancy required for each configuration process. The remainder of this chapter describes the requirements which shall be satisfied for all configuration tools that use an EDS. These requirements describe:

- EDS encoding


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- EDS command language
- EDS syntax

**Figure 7-3.4 Compliancy Requirements for the Configuration Process**

## 7-3.3 Using an EDS with Configuration Tools

CIP configuration tools:

- Extract user prompt information from a standard EDS
- Present this information to the user in a human-readable form

The tool developer determines the:

- user interface format
- operator interaction

### 7-3.3.1 EDS Interpreter Functions

The interpreter shall:

- Collect parameter selections required by an EDS
- Build the CIP messages necessary to configure a device
- Contain the object addresses for each device parameter requiring configuration


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.3.2 EDS File Organization

This section contains the file encoding requirements for EDSs. The EDS encoding requirements define the standard file encoding format to use for CIP products without regard to the configuration tool host platform or file system.

The term "file" as used in this chapter refers to any recognized file format associated with a configuration tool's file system without regard to the file storage media. An EDS contains an ASCII representation of a device's Parameter Objects and some additional information required to support object addressing.

#### 7-3.3.2.1 EDS Content

A single file must contain the entire EDS. The following table summarizes the structure of the sections, legal section delimiters, and the placement of sections in an EDS.

**Table 7-3.1 Sections of an EDS File**

| Section Name | Legal Delimiter | Placement | Required/Optional |
|---|---|---|---|
| File Description | [File] | 1 | Required |
| Device Description | [Device] | 2 | Required |
| I/O Characteristics | [IO_Info] | 3 or later | DeviceNet specific. See Volume 3, DeviceNet Adaptation of CIP for details |
| Enumerated Parameters | [EnumPar] | 3 or later | DeviceNet specific. See Volume 3, DeviceNet Adaptation of CIP for details |
| Parameters | [Params] | 3 or later | Optional |
| Parameter Groups | [Groups] | 3 or later | Optional |
| Device Classification | [Device Classification] | 3 or later | Required |
| Modular | [Modular] | 3 or later | Optional |
| Variant_IO_Info | [Variant_IO_Info] | 3 or later | DeviceNet specific. See Vol 3, DeviceNet Adaptation of CIP for details |
| Capacity | [Capacity] | 3 or later | Optional |
| Event Enumeration | [EventEnum] | 3 or later | Optional |
| Symbolic Translation | [SymbolicTranslation] | 3 or later | Conditional, see section 7-3.6.16 |
| Public object class sections | See section 7-3.6.1.1 | 3 or later | See section 7-3.6.1.1 |
| Vendor Specific object class sections | See section 7-3.6.1.2 | 3 or later | See section 7-3.6.1.2 |
| Internationalization | [Internationalization] | 3 or later | Optional |
| Modbus Mapper | [Modbus_Mapper] | 3 or later | Modbus specific. See Volume 7, Modbus Integration for details |

Observe these rules when defining an EDS:

- EDS White Space - The EDS interpreter shall treat certain characters as white space characters. These characters, read by the interpreter but not encoded as human-readable characters, designate the presence of blank space in a file. White space characters are:
  - New line
  - Carriage Return
  - Linefeed
  - Tabs, vertical and horizontal
  - Form Feed


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

    - End of File marker
    - Comments
- Keyword Characters – All keywords within an EDS file must be composed of ASCII characters from the following list:
    - Upper case letters A through Z
    - Lower case letters a through z
    - Numerals 0 through 9
    - The special character underscore "_"
    - The space character (The space may only be used in a section keyword. The space may only appear internal to a section name, and multiple sequential spaces are invalid)
- Sections - The EDS file must be partitioned into required and optional sections.
- Section delimiters - Each section in the EDS must be properly delimited by a *section keyword*.
- Section Keywords – Each section keyword is defined to be the text between the beginning of section keyword delimiter "[" and the terminating delimiter "]". The characters valid for use in section keywords are shown above. In addition, section keywords also allow the non-consecutive use of the characters "/" and "-". Section keywords shall not be case sensitive. There are two types of section keywords, public and vendor specific.
- Section order - Each required section must be placed in the required order.
- Entry - Each section in the EDS contains one or more *entries* beginning with an *entry keyword* followed by an equal sign. The entry keyword meaning depends on the context of the section. A semicolon indicates the end of the entry, and entries may span multiple lines.
- Entry Keywords – An entry keyword consists of a unique sequence of keyword characters. Entry keywords shall not be case sensitive. There are two types of entry keywords, public and vendor specific.
- Public Section and Entry Keyword – Public section and entry keywords shall always be defined by ODVA in Chapter 7 of one of the specification volumes. A public keyword shall never begin with a number.
- Entry fields - Each entry contains one or more fields. A comma delimiter separates each field. Optional fields not provided, referred to as empty fields, shall be one of the following:
    - A white space or nothing between commas (for example: keyword = x,,x;).
    - A white space or nothing between a comma and a semi-colon (for example: keyword = x,;).
    - A white space or nothing between an equal and comma (for example: keyword = ,x;).
    - A white space or nothing between an equal and semi-colon (for example: keyword = ;).
    - Trailing optional fields can be omitted by ending the entry with a semicolon. The meaning of the field(s) depends on the context of the section.
- Complex data fields – Certain entry fields must be specified with data that cannot be specified by a single value between the comma delimiters. The ability to further delimit an entry field is defined via the use of one or more sets of matching brace characters "{" and "}". The content between brace characters is considered a single item or entry. Content may be grouped in multiple braces.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- Vendor-Specific Keywords – Section, Entry and Field Keywords may be vendor-specific. Vendor Specific Section Keywords shall begin with the Vendor ID of the company making the addition followed by an underscore (*VendorID_VendorSpecificKeyword)*. Vendor Specific Entry Keywords not in a vendor-specific section shall begin with the Vendor ID of the company making the addition followed by an underscore (*VendorID_VendorSpecificKeyword*). Vendor Specific Entry Keywords in a vendor-specific section are always vendor-specific; they are not required to begin with the Vendor ID of the company making the addition. The VendorID shall be displayed in decimal and shall not contain leading zeroes. Each vendor is responsible for maintaining and documenting their vendor-specific keywords.

The following examples highlight the structure of the electronic data sheet (Note: The "$" character denotes a comment.)

**Figure 7-3.5 Basic Structure of an EDS file**

```
[section name]
$ Comment - extends to end of line

Entry1=Field1, Field2, Field3;          $ Entire entry on one line

Entry2=Field1, Field2, Field3, Field4;  $ Entire entry on one line

Entry3=       $ Multiple line entry
   Field1,                $ Field1
   Field2,                $ Field2
   Field3;                $ Field3

Entry4=       $ Combination
   Field1, Field2,        $ Fields 1 and 2 on one line
   Field3,                $ Field3
   Field4;                $ Field4

65535_Entry=  $ Vendor Specific entry for Vendor_ID 65535
   Field1, Field2;        $ with two fields

Entry5 = 1,               $ Field 1 specifies the value 1
  {1,2,3};                $ Field 2 specifies an array or
                          $ structure with three values

Entry6 = { 44, {22,33,11} };    $ Entry 6 specifies a single field.
                                $ The field contains two sets of data.
                                $ The first set is a single value 44.
                                $ The second set contains three values.
```

Note from the examples that an entry can extend over multiple lines as long as commas properly delimit the fields. The configuration tool ignores any whitespace characters, including comments, tabs, and spaces. Comments start at the comment delimiter ($) and extend to the end of the line. All entries must terminate with a semicolon.

The configuration tool shall ignore any entry keyword that contains a field value the configuration tool does not recognize.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.4 File Naming Requirements

Currently no file naming conventions exist for disk-based EDS files, except for files in a DOS/Windows environment. The file should have the suffix ".EDS" appended to the file name.

### 7-3.5 EDS Data Encoding Requirements

All data in the EDS must be encoded using 8-bit ASCII characters, where all references to "ASCII characters" mean an 8-bit ASCII character format (as defined by Tables 1 and 2, Row 00 of the IEC/ISO 10646-1). Characters that cannot be displayed on an ANSI terminal cannot be used in identifier names or in data representations.

#### 7-3.5.1 Elementary Data Type Encodings

##### 7-3.5.1.1 Character String Conventions

###### 7-3.5.1.1.1 STRING, SHORT STRING – Character String (1 Byte per Character)

All references to STRING and SHORT STRING in this chapter are represented as ASCII character strings of fixed length, without null terminators and enclosed in double quotes.

Characters contained between double quotation marks are converted into 8-bit ASCII codes. The following is an example:

- "This results in a string composed by 8-bit characters".

###### 7-3.5.1.1.2 STRING2 – Character String (2 Bytes per Character)

All references to STRING2 in this chapter are represented as ASCII character strings of fixed length, without null terminators, preceded by a capital L and enclosed in double quotes.

Characters contained between double quotation marks are converted into UNICODE (16-bit) codes. The following is an example:

- L"A string of UNICODE characters, including the Greek character Pi \u03C0"

Note: the text \u03C0 specifies a single 16-bit character whose value is 0x03C0. In the UNICODE character set, this is Table 9, Row 3, Basic Greek – the character for lower case "Pi". Descriptions of character escape sequences are described in subsequent section 7-3.5.1.1.8.

###### 7-3.5.1.1.3 STRINGN – Character String (N-Bytes per Character)

There is no definition for STRINGN EDS representation.

###### 7-3.5.1.1.4 STRINGI – International Character Strings

The CIP International String (STRINGI) data follows the format as defined in Appendix C. The STRINGI is specified in an EDS file as a complex data representation with the structure described below.

```
{Number,
{LanguageChars, CharStringStruct, CharSet, InternationalString},
…
}
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 7-3.2 STRINGI Field Definitions**

| Field Name | Field Number | Data Type | Definition |
|---|---|---|---|
| Number | 1 | USINT | The number of internationalized character strings |
| LanguageChars | 2, 6, … | STRING | The ISO 639-2T language. The LanguageChars for the language come from ISO 639-2/T (Alpha-3 Terminology Code). Some common language codes are shown in Table C-2.2. |
| CharStringStruct | 3, 7, … | USINT | The structure of the character string, limited to the values 0xD0, 0xD5 and 0xDA |
| CharSet | 4, 8, … | UINT | The character set which the character sting is based on. The character set comes from IANA MIB Printer Codes (RFC 1759). The character set values are limited to those values that are provided in Table C-2.3 |
| InternationalString | 5, 9, … | STRING, STRING2 | The actual international character string |

For example, the following represents a STRINGI specification with three languages:

Field1 = { 3,

{"eng",0xD0,4,"This is an ASCII English language string"},
{"spa",0xD5,1000,L"Españoles palabras"}, $ “Spanish words” using UNICODE
{"deu",0xD0,4, "Spanische Wörter auf Deutsch"} $ “Spanish words in German”

};

### 7-3.5.1.1.5 Handling Insufficient Characters in a String Field

An EDS interpreter uses right-justification of characters in a field and fills any unspecified characters with leading blanks (ASCII 0x20) for the remaining length of the string. For example, if a parameter has a maximum string length of 8 and receives the string

”123AB”,

the interpreter decodes the string as:

”~~~123AB”,

where the tilde characters (~) represent spaces.

### 7-3.5.1.1.6 Handling Excess Characters in a String Field

If a given string field contains too many characters, the EDS interpreter truncates characters from left to right. For example, if a parameter has a maximum string length of 8 and receives the string

”I23ABCDEFG”,

the EDS interpreter truncates the string to

”I23ABCDE”.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.5.1.1.7 String Concatenation

Multiple strings with no intervening commas are concatenated. For example, the line

”ABC” ”123” ”XYZ”

is interpreted as:

”ABC123XYZ”

The strings may also be on separate lines. For example, the following lines

”ABC” $ this is a comment

”123”

”XYZ”

are also interpreted as:

”ABC123XYZ”

For a UNICODE string (long string), only the first double quotation mark shall be preceded by a capital L. Example: L”ABC” ”123” ”XYZ” is the same as L”ABC123XYZ”.

#### 7-3.5.1.1.8 String Escape Sequences

The EDS interpreter shall recognize all listed sequences. Interpretation is application specific. The escape sequences are:

\\ translates to a \

\n translates to a newline

\t translates to a tab

\v translates to a vertical tab

\b translates to a backspace

\r translates to a carriage return

\f translates to a form feed

\a translates to the BELL character (0x07)

\? translates to a ?

\0 translates to a null

\” translates to a ”

\’ translates to ’

\xnn translates to a single byte containing the value of “nn” as expressed in hexadecimal

\unnnn translates to a pair of bytes containing the value of “nnnn”as expressed in hexadecimal. This form of string escape is only valid where the resultant string data is 16-bits in length, e.g. the L" form of string specification.

If a sequence not listed above is encountered, the interpreting device shall reject the entire string and indicate an error. EDS files shall only contain escape sequences defined above.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.5.1.2 BOOL, USINT, UINT, UDINT, ULINT – Unsigned Integer Value Convention

The unsigned integer data types represent positive integer values. Unsigned integer data may be entered in decimal or hexadecimal notation with no white space or commas between characters. If hexadecimal notation represents the unsigned integer characters, the two character sequence 0x, with no white space, shall precede the unsigned integer characters.

The range of legal **BOOL** data is:

| | | | |
|---|---|---|---|
| Decimal Notation: | 0 | to | 1 |
| Hexadecimal Notation: | 0x0 | to | 0x1 |

The range of legal **USINT** data is:

| | | | |
|---|---|---|---|
| Decimal Notation: | 0 | to | 255 |
| Hexadecimal Notation: | 0x0 | to | 0xFF |

The range of legal **UINT** data is:

| | | | |
|---|---|---|---|
| Decimal Notation: | 0 | to | 65535 |
| Hexadecimal Notation: | 0x0 | to | 0xFFFF |

The range of legal **UDINT** data is:

| | | | |
|---|---|---|---|
| Decimal Notation: | 0 | to | 4294967295 |
| Hexadecimal Notation: | 0x0 | to | 0xFFFFFFFF |

The range of legal **ULINT** data is:

| | | | |
|---|---|---|---|
| Decimal Notation: | 0 | to | 18446744073709551615 |
| Hexadecimal Notation: | 0x0 | to | 0xFFFFFFFFFFFFFFFF |

#### 7-3.5.1.3 SINT, INT, DINT, LINT – Signed Integer Value Convention

The SINT, INT, DINT and LINT data types represent signed integer data values. Signed integer data may be entered in decimal or hexadecimal notation with no white space or commas between characters. If hexadecimal notation represents the signed integer characters, the two character sequence 0x, with no white space, shall precede the integer value characters.

The range of legal **SINT** data is:

| | | | |
|---|---|---|---|
| Decimal Notation: | -128 | to | 127 |
| Hexadecimal Notation: | 0x80 | to | 0x7F |

The range of legal **INT** data is:

| | | | |
|---|---|---|---|
| Decimal Notation: | -32768 | to | 32767 |
| Hexadecimal Notation: | 0x8000 | to | 0x7FFF |

The range of legal **DINT** data is:

| | |
|---|---|
| Decimal Notation: | -2147483648 to 2147483647 |
| Hexadecimal Notation: | 0x80000000 to 0x7FFFFFFF |

The range of legal **LINT** data is:

| | |
|---|---|
| Decimal Notation: | -9223372036854775808 to 9223372036854775807 |
| Hexadecimal Notation: | 0x8000000000000000 to 0x7FFFFFFFFFFFFFFF |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.5.1.4 BYTE, WORD, DWORD, LWORD – Bit String Convention

The BYTE, WORD, DWORD and LWORD data types represent bit-addressable values. These values are considered discrete bit position values and are not intended to represent either signed or unsigned integer values. However, these values, for convenience, may be entered in decimal, hexadecimal or binary notation with no white space or commas between characters. If hexadecimal notation represents value characters, the two character sequence 0x, with no white space, shall precede the value characters.

The range of legal **BYTE** data is:

| | | | |
|---|---|---|---|
| Decimal Notation: | 0 | to | 255 |
| Hexadecimal Notation: | 0x0 | to | 0xFF |
| Binary Notation: | 0b00000000 to 0b11111111 | | |

The range of legal **WORD** data is:

| | | | |
|---|---|---|---|
| Decimal Notation: | 0 | to | 65535 |
| Hexadecimal Notation: | 0x0 | to | 0xFFFF |
| Binary Notation: | 0b0000000000000000 to 0b1111111111111111 | | |

The range of legal **DWORD** data is:

| | | | |
|---|---|---|---|
| Decimal Notation: | 0 | to | 4294967295 |
| Hexadecimal Notation: | 0x0 | to | 0xFFFFFFFF |
| Binary Notation: | 0b00000000000000000000000000000000 to 0b11111111111111111111111111111111 | | |

The range of legal **LWORD** data is:

| | | | |
|---|---|---|---|
| Decimal Notation: | 0 | to | 18446744073709551615 |
| Hexadecimal Notation: | 0x0 | to | 0xFFFFFFFFFFFFFFFF |
| Binary Notation: | 0b0000000000000000000000000000000-0000000000000000000000000000000 to 0b1111111111111111111111111111111-1111111111111111111111111111111 | | |

### 7-3.5.1.5 REAL, LREAL – Floating Point Value Convention

The REAL and LREAL data types represent binary floating-point values. The internal representation of these data formats is described by the IEEE Standard 754. This standard describes both numeric values and bit sequences which are interpreted as “Not a Number” (NaN) symbolic values and positive and negative infinity. The floating point values can be entered as either integer values, values based upon decimal floating point representation, or values entered in “scientific” notation using a mantissa and an exponent.

The range of legal REAL values for each of these formats that may appear in an EDS file are:

Integer (Fixed) Notation: -16777216 to 16777216

These min/max values were chosen because they are the limits that can be placed into the 23-bit mantissa (in combination with the implied bit 23 set to 1). This allows exact representation of all the integers within this range.

Decimal (Floating Point) Notation: 0.0 to ±999999999999999.9


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Decimal floating-point values are those, which have both a whole integer and fractional component. The integer and fractional components are separated by a decimal point. There must be at least one digit on either side of the decimal point. An arbitrary limit of 16 digits has been chosen.

Scientific Notation: ±n.nnnnnnnnnnE±xx

The scientific notation form of the value is made up of a mantissa and exponent, separated by an E. The exponent (xx) is always a signed integer power of ten (E) applied to the mantissa (n.nnnnnnnnnn).

The total number of digits in the mantissa shall not exceed 11. There must be at least one digit on either side of the decimal point. The number of digits in the exponent shall not exceed 2.

REAL values outside the range of ±3.4028235e+38 cannot be represented. REAL values smaller in magnitude than ±1.4012985e-45 cannot be represented.

The range of legal LREAL values for each of these formats that may appear in an EDS are:

Integer (Fixed) Notation: -9007199254740992 to 9007199254740992

These min/max values were chosen because they are the limits that can be placed into the 52-bit mantissa (in combination with the implied bit 52 set to 1). This allows exact representation of all the integers within this range.

Decimal (Floating Point) Notation: 0.0 to ±999999999999999.9

Decimal floating-point values are those, which have both a whole integer and fractional component. The integer and fractional components are separated by a decimal point. There must be at least one digit on either side of the decimal point. An arbitrary limit of 16 digits has been chosen.

Scientific Notation: ±n.nnnnnnnnnnnnnnnnnE±xxx

The scientific notation form of the value is made up of a mantissa and exponent, separated by an E. The exponent (xxx) is always a signed integer power of ten (E) applied to the mantissa (n.nnnnnnnnnnnnnnnnn).

The total number of digits in the mantissa shall not exceed 18 and the number of digits in the exponent shall not exceed 3.

LREAL values outside the range of ±1.79769313486231570E+308 cannot be represented. LREAL values smaller in magnitude than ±4.94065645841246544E-324 cannot be represented.

In addition to the above value entries, the floating-point representation also allows for two styles of “Not a Number” (NaN) symbolic entries and positive and negative infinity. For these cases, the following special words are reserved and shall be used to represent the entry of the associated floating-point symbol:

- Quiet Not a Number: QUIET-NAN
- Signaling Not a Number: SIGNAL-NAN
- Positive Infinity: INFINITY (or +INFINITY)
- Negative Infinity: -INFINITY

#### 7-3.5.1.6 STIME – Synchronous Time convention

There is no definition for STIME representation.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.5.1.7 DATE – Date convention

The DATE data type represents date information values. DATE data may be entered in formatted, decimal or hexadecimal notation with no white space or commas between characters. If hexadecimal notation represents the DATE, the two character sequence 0x, with no white space, shall precede the integer value characters.

The range of legal DATE data is:

| | |
|---|---|
| Formatted Notation: | yyyy-mm-dd (yyyy – year, mm – month, dd – day) – 1972-01-01 to 2151-06-06 |
| Decimal Notation: | 0 to 65535 |
| Hexadecimal Notation: | 0x0000 to 0xFFFF |

#### 7-3.5.1.8 TIME_OF_DAY – Time of Day convention

The TIME_OF_DAY data type represents time of day information values. TIME_OF_DAY data may be entered in formatted, decimal or hexadecimal notation with no white space or commas between characters. If hexadecimal notation represents the TIME_OF_DAY, the two character sequence 0x, with no white space, shall precede the integer value characters.

The range of legal TIME_OF_DAY data is:

| | |
|---|---|
| Formatted Notation: | hh:mm:ss.iii (hh – hour, mm – minute, ss – second, iii – millisecond) – 00:00:00.000 to 23:59:59.999 |
| Decimal Notation: | 0 to 86399999 |
| Hexadecimal Notation: | 0x00000000 to 0x05265BFF |

#### 7-3.5.1.9 DATE_AND_TIME – Date and Time of Day convention

The DATE_AND_TIME data type represents date and time of day information values. DATE_AND_TIME data may be entered in formatted, decimal or hexadecimal notation with no white space or commas between characters. If hexadecimal notation represents the DATE_AND_TIME, the two character sequence 0x, with no white space, shall precede the integer value characters.

The range of legal DATE_AND_TIME data is:

| | |
|---|---|
| Formatted Notation: | yyyy-mm-dd-hh:nn:ss.iii (yyyy – year, mm – month, dd – day, hh – hour, nn – minute, ss – second, iii – millisecond) – 1972-01-01-00:00:00.000 to 2151-06-06-23:59:59.999 |
| Decimal Notation: | {0, 0} to {86399999, 65535} |
| Hexadecimal Notation: | {0x00000000, 0x0000} to {0x05265BFF, 0xFFFF} |

#### 7-3.5.1.10 FTIME – Duration (high resolution) convention

The FTIME data type represents duration (high resolution) information values. FTIME data may be entered in formatted, decimal or hexadecimal notation with no white space or commas between characters. If hexadecimal notation represents the FTIME, the two character sequence 0x, with no white space, shall precede the integer value characters.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The range of legal FTIME data is:

| | |
|---|---|
| Formatted Notation: | xxmyy.yyyyyys (xx – minutes, yy.yyyyyy - seconds) – -35m47.483648s to 35m.47.483647s |
| Decimal Notation: | -2147483648 to 2147483647 |
| Hexadecimal Notation: | 0x80000000 to 0x7FFFFFFF |

#### 7-3.5.1.11 LTIME – Duration (long) convention

The LTIME data type represents duration (long) information values. LTIME data may be entered in formatted, decimal or hexadecimal notation with no white space or commas between characters. If hexadecimal notation represents the LTIME, the two character sequence 0x, with no white space, shall precede the integer value characters.

The range of legal LTIME data is:

| | |
|---|---|
| Formatted Notation: | aaaaaaaaadbbhccmdd.dddddds (aaaaaaaaa – days, bb – hours, cc – minutes, dd.dddddd - seconds) – -106751991d4h0m54.775808s to 106751991d4h0m.54.775807s |
| Decimal Notation: | -9223372036854775808 to 9223372036854775807 |
| Hexadecimal Notation: | 0x8000000000000000 to 0x7FFFFFFFFFFFFFFF |

#### 7-3.5.1.12 ITIME – Duration (short) convention

The ITIME data type represents duration (short) information values. ITIME data may be entered in formatted, decimal or hexadecimal notation with no white space or commas between characters. If hexadecimal notation represents the ITIME, the two character sequence 0x, with no white space, shall precede the integer value characters.

The range of legal ITIME data is:

| | |
|---|---|
| Formatted Notation: | xxsyyyms (xx – seconds, yyy – millseconds) – -32s768ms to 32s767ms |
| Decimal Notation: | -32768 to 32767 |
| Hexadecimal Notation: | 0x8000 to 0x7FFF |

#### 7-3.5.1.13 TIME – Duration (milliseconds) convention

The TIME data type represents duration (milliseconds) information values. TIME data may be entered in formatted, decimal or hexadecimal notation with no white space or commas between characters. If hexadecimal notation represents the TIME, the two character sequence 0x, with no white space, shall precede the integer value characters.

The range of legal TIME data is:

| | |
|---|---|
| Formatted Notation: | xxdyyhzzmaa.aaas (xx – days, yy – hours, zz – minutes, aa.aaa – seconds) – -24d20h31m23.648s to 24d20h31m23.647s |
| Decimal Notation: | -2147483648 to 2147483647 |
| Hexadecimal Notation: | 0x80000000 to 0x7FFFFFFF |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.5.1.14 EPATH – Path Format Convention

A path is represented in an EDS file as groups of two adjacent hexadecimal characters separated by spaces. Double quotes enclose the entire string of characters. EPATH data types specified as EDS keyword field values will specify if the EPATH is packed or padded (See section C-1.3 for the definition of packed and padded). Some example EPATHs are:

Padded EPATH: “20 04 25 00 01 01 30 03”
Packed EPATH: “20 04 25 01 01 30 03”

The EPATH may include references to ParamN or ProxyParamN (Param) keywords (see section 7-3.6.5 for the definition of the ProxyParamN and ParamN keywords). This reference shall be made by specifying the name of the Param keyword in the EPATH. Param entries in EPATHs shall evaluate to a USINT, UINT or UDINT. The resulting EPATH shall conform to the EPATH definition in Appendix C-1.3. The value of the Param shall be used in a little endian order for insertion into the EPATH. The Param references within the EPATH may be enclosed in brackets as shown in the examples below.

EPATH with Param #1: “20 04 24 [Param1] 30 03”
EPATH with ProxyParam #2: “20 04 24 [ProxyParam2] 30 03”

When enclosed in brackets, the value of the Param shall be local to the path – the same Param entry may have a different value elsewhere in the EDS. If the Param is not enclosed in brackets, the value shall be the same everywhere within the EDS.

The Param keyword in an EPATH is limited to the Logical Value of a Logical Segment. The configuration software shall set the Logical Format of the Logical Segment to indicate an 8 bit, 16 bit or 32 bit Logical Value depending on the number of bits it takes to represent the Logical Value. If the EPATH is a padded EPATH the configuration software shall insert a 00 pad byte after the segment type byte if the Logical Format indicates a 16 or 32 bit Logical Value.

The user interface may ask the user to specify a value for Param1 in the example above. If the user specifies “5” the resulting EPATH would be “20 04 24 05 30 03”.

The EPATH may also include the keyword, SYMBOL_ANSI. When SYMBOL_ANSI in encountered it shall evaluate to an extended symbolic segment (see Appendix C) entered through the user interface. The extended symbol segment shall be an ANSI extended symbol (CIP path type = 0x91). For example, the string "CAB" shall evaluate to the following extended symbol segment (padded): "91 03 43 41 42 00". See below for an example of SYMBOL_ANSI EPATH usage:

EPATH with symbol: “SYMBOL_ANSI”

The user interface shall ask the user to specify a symbol string (like “CAB”) which results in an EPATH of “91 03 43 41 42 00”.

#### 7-3.5.1.15 Array of EPATHs – Array of Paths Format Convention

An array of paths is represented in an EDS file as a complex data field containing one or more EPATHs. Some examples of an array containing 2 EPATHs are:

Array of Padded EPATH: {“20 04 25 00 01 01 30 03”, “20 04 25 00 01 01 30 04”}
Array of Packed EPATH: {“20 04 25 01 01 30 03”, “20 04 25 01 01 30 04”}


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.5.1.16 ENGUNIT – Engineering Units convention

An engineering unit is represented in an EDS file the same as a UINT. For example, the engineering unit value for meters per second (see Appendix D, Table D-2.30) would be entered as 0x2B00 or 11008.

#### 7-3.5.1.17 EDS_External_Id – Non-CIP Modular Module Identifier

An External Id is represented in an EDS file as groups of two adjacent hexadecimal characters separated by spaces. Double quotes enclose the entire string of characters. The number of hexadecimal pairs can range between 1 and 16. Some example External Ids are:

"00"
"36 AB 9F"
"00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF"

The value of an External Id is treated as a byte stream with vendor specific meaning.

### 7-3.5.2 Defined Constructed Data Type Encodings

#### 7-3.5.2.1 SIGNED_ODOMETER – Signed large magnitude counters

The SIGNED_ODOMETER type represents a signed large magnitude counter. SIGNED_ODOMETER data may be entered in decimal notation with no white space or commas between characters.

The range of legal SIGNED_ODOMETER data is:

Decimal Notation: {-999,-999,-999,-999,-999} to {999,999,999,999,999}

The decimal point location of the SIGNED_ODOMETER is determined by defining the exponent value in the Type Specific Field of the ConstructedParamN keyword.

The ConstructedParamN Type Specific field information is:

- The Type Specific Field Count (field 13) value is 1.
- The Type Specific Field (field 14) type is SINT. The value of this field is the exponent "n", see section C-2.1.5.1 for the definition of the SIGNED_ODOMETER constructed data type

#### 7-3.5.2.2 ODOMETER – Large magnitude counters

The ODOMETER type represents a large magnitude counter. ODOMETER data may be entered in decimal notation with no white space or commas between characters.

The range of legal ODOMETER data is:

Decimal Notation: {0, 0, 0, 0, 0} to {999, 999, 999, 999, 999}

The decimal point location of the ODOMETER is determined by defining the exponent value in the Type Specific Field of the ConstructedParamN keyword.

The ConstructedParamN Type Specific field information is:

- The Type Specific Field Count (field 13) value is 1.
- The Type Specific Field (field 14) type is SINT. The value of this field is the exponent "n", see section C-2.1.5.2 for the definition of the ODOMETER constructed data type.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.5.3 EDS-specific Data Type Encodings

#### 7-3.5.3.1 EDS_Date Convention

The EDS_Date data type shall be of the format; mm-dd-yyyy, where mm is the month, dd is the day of the month and yyyy is the year. Valid values for the month, day and year portions of the mm-dd-yyyy shall be

- mm 01 through 12
- dd 01 through 31 (depending upon the month and year)
- yyyy 1994 through 9999.

Two character years representations may be used in which case the EDS_Date data type shall be of the format; mm-dd-yy, where mm is the month, dd is the day of the month and yy is the year. In this case, the two digits for the year have an implied leading 19, such that yy=96 shall represent the year 1996. Valid values for the month, day and year portions of the mm-dd-yy parameters shall be:

- mm 01 through 12
- dd 01 through 31 (depending upon the month and year)
- yy 94 through 99 (Note that a leading 19 is implied)

NOTE: Two character year representations are not recommended.

#### 7-3.5.3.2 EDS_Time_Of_Day Convention

The EDS_Time_of_Day data type shall be of the format; hh:mm:ss, where hh is hours, mm is minutes and ss is seconds. Valid values for the hours, minutes and seconds shall be:

- hh 00 through 23
- mm 00 through 59
- ss 00 through 59

#### 7-3.5.3.3 EDS_URL – Uniform Resource Locator

All references to EDS_URL within this specification are for the formalized information necessary to locate and access resources via an internet capable mechanism. Double quotes enclose all EDS_URLs. The form of the EDS_URL is that defined by the internet's Network Working Group RFC1738 "Uniform Resource Locator (URL)". For specifications made within an EDS file, the EDS_URL is limited to any of the forms:

- http
- ftp
- file

#### 7-3.5.3.4 EDS_Revision – Revision

The revision must be formatted as:

major_revision.minor_revision

For example: 2.1, where major_revision is 2, and minor_revision is 1

The data type of major_revision and minor_revision shall be USINT.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.5.3.5 EDS_Field_Keyword - Field Keyword Convention

A field keyword is represented in an EDS as text with no blanks. When an entry keyword lists the data type of a field as EDS_Field_Keyword, the specific keywords allowed shall be listed. A vendor specific field keyword shall only be allowed if specified in the field definition. If there are only a few allowed values, they may be included in a table as shown in the example below:

| Field Name | Field Number | Data Type | Required/Options |
|---|---|---|---|
| … | | | |
| … | … | EDS_Field_Keyword, allowed values: NONE, WRITE, READ_WRITE | … |
| … | | | |

#### 7-3.5.3.6 EDS_Service – Service Keyword Convention

EDS_Service is formatted as:

- USINT – Indication of a non-attribute specific service code as defined in Table A-3.1
- {USINT, UINT, UINT, UINT, …} – Indication of a service code per Table A-3.1 and associated attribute identifiers to which the service is accepted.
- {USINT, *} – Indication of a service code per Table A-3.1 that is associated with every attribute listed in the corresponding Class_Attributes/Instance_Attribute list.

#### 7-3.5.3.7 EDS Entry Keyword Reference Convention

The fields of some EDS Entry Keywords may allow other EDS Entry Keywords to be referenced. The EDS Keywords that are valid reference types are:

ArrayN
AssemN
BitStringVariantN
ProxyAssemN
ProxiedAssemN
AssemExaN
ProxyAssemExaN
ProxiedAssemExaN
ParamN
ProxyParamN
ProxiedParamN
VariantN
VariantExaN

When the data type of an EDS Entry Keyword field is one of the EDS Entry Keyword references (in the list above) the ending N shall be replaced by the integer value of the actual EDS Entry Keyword instance, e.g. Param5.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.5.4 EDS Comments

Comments must be delimited with the dollar sign character ($) and the new line character. The EDS interpreter treats all characters between the comment delimiters as white space. Some example comments are:

$ This is a valid comment line <NL>

1, 2, 3; $ This is a valid comment <NL>

$ Comments cannot span <NL>
more than one line <NL> <= This is an error - no $

## 7-3.6 Basic EDS File Requirements

This section describes each section of an EDS and specifies the usage requirements.

**Table 7-3.3 Locating EDS Requirements**

| For EDS requirements for: | Go to section: |
|---|---|
| Public Object Class Sections | 7-3.6.1.1 |
| Vendor Specific Object Class Sections | 7-3.6.1.2 |
| File Description Section | 7-3.6.2 |
| Device Description Section | 7-3.6.3 |
| Device Classification Section | 7-3.6.4 |
| Parameter Class Section | 7-3.6.5 |
| Parameters Section | 7-3.6.6 |
| Parameter Groups Section | 7-3.6.7 |
| Assembly Section | 7-3.6.8 |
| Connection Manager Section | 7-3.6.10 |
| Port Section | 7-3.6.12 |
| Capacity Section | 7-3.6.13 |
| Connection Configuration Class Section | 7-3.6.14 |
| Event Enumeration Section | 7-3.6.15 |
| Symbolic Translation Section | 7-3.6.16 |
| Internationalization Section | 7-3.6.17 |
| Modular Section | 7-3.7.1 |

### 7-3.6.1 Common Object Class Information

The inclusion of a Common Object Class section in an EDS file indicates the devices supports at least the minimum implementation of the object. Because not all clients have native support for all objects it is recommended that entries (ParamN or AssemN) for object attributes be included as appropriate.

#### 7-3.6.1.1 Public Object Class Sections

The following table identifies section keywords that are used to publicize public object class information. Each of the following Public Object Class Sections may contain any of the Common Object Class Entry Keyword(s) described in section 7-3.6.1.3. The following table contains Public Object Class Section names.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 7-3.4 Public Object Class Section Keywords**

| CIP Object | Section Keyword Name | Required/Optional | See section |
|---|---|---|---|
| Identity Object<br>01 hex | [Identity Class] | Optional | |
| Message Router Object<br>02 hex | [Message Router Class] | Optional | |
| DeviceNet Object<br>03 hex | [DeviceNet Class] | See Chapter 7 of Volume 3, DeviceNet Adaptation of CIP | |
| Assembly Object<br>04 hex | [Assembly] | Optional | 7-3.6.8 |
| Connection Object<br>05 hex | [Connection Class] | Optional | |
| Connection Manager Object<br>06 hex | [Connection Manager] | Optional | 7-3.6.10 |
| Register Object<br>07 hex | [Register Class] | Optional | |
| Discrete Input Point Object<br>08 hex | [Discrete Input Class] | Optional | |
| Discrete Output Point Object<br>09 hex | [Discrete Output Class] | Optional | |
| Analog Input Point Object<br>0A hex | [Analog Input Class] | Optional | |
| Analog Output Point Object<br>0B hex | [Analog Output Class] | Optional | |
| Presence Sensing Object<br>0E hex | [Presence Sensing Class] | Optional | |
| Parameter Object<br>0F hex | [ParamClass] | Optional | 7-3.6.5 |
| Parameter Group Object<br>10 hex | [Groups] | Optional | 7-3.6.7 |
| Group Object<br>12 hex | [Group Class] | Optional | |
| Discrete Input Group Object<br>1D hex | [Discrete Input Group Class] | Optional | |
| Discrete Output Group Object<br>1E hex | [Discrete Output Group Class] | Optional | |
| Discrete Group Object<br>1F hex | [Discrete Group Class] | Optional | |
| Analog Input Group Object<br>20 hex | [Analog Input Group Class] | Optional | |
| Analog Output Group Object<br>21 hex | [Analog Output Group Class] | Optional | |
| Analog Group Object<br>22 hex | [Analog Group Class] | Optional | |
| Position Sensor Object<br>23 hex | [Position Sensor Class] | Optional | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| CIP Object | Section Keyword Name | Required/Optional | See section |
|---|---|---|---|
| Position Controller Supervisor Object<br>24 hex | [Position Controller Supervisor Class] | Optional | |
| Position Controller Object<br>25 hex | [Position Controller Class] | Optional | |
| Block Sequencer Object<br>26 hex | [Block Sequencer Class] | Optional | |
| Command Block Object<br>27 hex | [Command Block Class] | Optional | |
| Motor Data Object<br>28 hex | [Motor Data Class] | Optional | |
| Control Supervisor Object<br>29 hex | [Control Supervisor Class] | Optional | |
| AC/DC Drive Object<br>2A hex | [AC/DC Drive Class] | Optional | |
| Acknowledge Handler Object<br>2B hex | [Acknowledge Handler Class] | Optional | |
| Overload Object<br>2C hex | [Overload Class] | Optional | |
| Softstart Object<br>2D hex | [Softstart Class] | Optional | |
| Selection Object<br>2E hex | [Selection Class] | Optional | |
| S-Device Supervisor Object<br>30 hex | [S-Device Supervisor Class] | Optional | |
| S-Analog Sensor Object<br>31 hex | [S-Analog Sensor Class] | Optional | |
| S-Analog Actuator Object<br>32 hex | [S-Analog Actuator Class] | Optional | |
| S-Single Stage Controller Object<br>33 hex | [S-Single Stage Class] | Optional | |
| S-Gas Calibration Object<br>34 hex | [S-Gas Calibration Class] | Optional | |
| Trip Point Object<br>35 hex | [Trip Point Class] | Optional | |
| Drive Data Object<br>N/A | [Drive Data Class] | Optional | |
| File Object<br>37 hex | [File Class] | Optional | |
| S-Partial Pressure Object<br>38 hex | [S-Partial Pressure Class] | Optional | |
| Safety Supervisor Object<br>39 hex | [Safety Supervisor Class] | See Chapter 7 of Volume 5, CIP Safety | |
| Safety Validator Object<br>3A hex | [Safety Validator Class] | See Chapter 7 of Volume 5, CIP Safety | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| CIP Object | Section Keyword Name | Required/Optional | See section |
|---|---|---|---|
| Safety Discrete Output Point Object<br>3B hex | [Safety Discrete Output Point Class] | See Chapter 7 of Volume 5, CIP Safety | |
| Safety Discrete Output Group Object<br>3C hex | [Safety Discrete Output Group Class] | See Chapter 7 of Volume 5, CIP Safety | |
| Safety Discrete Input Point Object<br>3D hex | [Safety Discrete Input Point Class] | See Chapter 7 of Volume 5, CIP Safety | |
| Safety Discrete Input Group Object<br>3E hex | [Safety Discrete Input Group Class] | See Chapter 7 of Volume 5, CIP Safety | |
| Safety Dual Channel Output Object<br>3F hex | [Safety Dual Channel Output Class] | See Chapter 7 of Volume 5, CIP Safety | |
| S-Sensor Calibration Object<br>40 hex | [S-Sensor Calibration Class] | Optional | |
| Event Log Object<br>41 hex | [Event Log Class] | Optional | |
| Motion Axis Object<br>42 hex | [Motion Axis Class] | Optional | |
| Time Sync Object<br>43 hex | [Time Sync Class] | Optional | |
| Modbus Object<br>44 hex | [Modbus Class] | See Chapter 7 of Volume 7, Integration of Modbus Devices into the CIP Architecture | |
| Originator Connection List Object<br>45 hex | [Originator Connection List Class] | Optional | |
| Modbus Serial Link Object<br>46 hex | [Modbus Serial Link Class] | See Chapter 7 of Volume 7, Integration of Modbus Devices into the CIP Architecture | |
| Device Level Ring Object<br>47 hex | [DLR Class] | See Chapter 7 of Volume 2, EtherNet/IP Adaptation of CIP | |
| QoS Object<br>48 hex | [QoS Class] | See Chapter 7 of Volume 2, EtherNet/IP Adaptation of CIP | |
| Safety Analog Input Point Object<br>49 hex | [Safety Analog Input Point Class] | See Chapter 7 of Volume 5, CIP Safety | |
| Safety Analog Input Group Object<br>4A hex | [Safety Analog Input Group Class] | See Chapter 7 of Volume 5, CIP Safety | |
| Safety Dual Channel Analog Input Object<br>4B hex | [Safety Dual Channel Analog Input Class] | See Chapter 7 of Volume 5, CIP Safety | |
| SERCOS III Link Object<br>4C hex | [SERCOS III Link Class] | See Chapter 7 of Volume 5, CIP Safety | |
| Target Connection List Object<br>4D hex | [Target Connection List Class] | Optional | |
| Base Energy Object<br>4E hex | [Base Energy Class] | Optional | |
| Electrical Energy Object<br>4F hex | [Electrical Energy Class] | Optional | |
| Non-electrical Energy Object<br>50 hex | [Non-electrical Energy Class] | Optional | |
| Base Switch Object<br>51 hex | [Base Switch Class] | See Chapter 7 of Volume 2, EtherNet/IP Adaptation of CIP | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| CIP Object | Section Keyword Name | Required/Optional | See section |
|---|---|---|---|
| SNMP Object<br>52 hex | [SNMP Class] | See Chapter 7 of Volume 2, EtherNet/IP Adaptation of CIP | |
| Power Management Object<br>53 hex | [Power Management Class] | Optional | |
| RSTP Bridge Object<br>54 hex | [RSTP Bridge Class] | See Chapter 7 of Volume 2, EtherNet/IP Adaptation of CIP | |
| RSTP Port Object<br>55 hex | [RSTP Port Class] | See Chapter 7 of Volume 2, EtherNet/IP Adaptation of CIP | |
| Parallel Redundancy Protocol Object<br>56 hex | [PRP Class] | See Chapter 7 of Volume 2, EtherNet/IP Adaptation of CIP | |
| PRP Nodes Table Object<br>57 hex | [PRP Nodes Table Class] | See Chapter 7 of Volume 2, EtherNet/IP Adaptation of CIP | |
| ControlNet Object<br>F0 hex | [ControlNet Class] | See Chapter 7 of Volume 4, ControlNet Adaptation of CIP | |
| ControlNet Keeper Object<br>F1 hex | [Keeper] | See Chapter 7 of Volume 4, ControlNet Adaptation of CIP | |
| ControlNet Scheduling Object<br>F2 hex | [Scheduling] | See Chapter 7 of Volume 4, ControlNet Adaptation of CIP | |
| Connection Configuration Object<br>F3 hex | [Connection Configuration] | Conditional [1] | 7-3.6.14 |
| Port Object<br>F4 hex | [Port] | Conditional [2] | 7-3.6.12 |
| TCP/IP Interface Object<br>F5 hex | [TCP/IP Interface Class] | See Chapter 7 of Volume 2, EtherNet/IP Adaptation of CIP | |
| Ethernet Link Object<br>F6 hex | [Ethernet Link Class] | See Chapter 7 of Volume 2, EtherNet/IP Adaptation of CIP | |
| CompoNet Link Object<br>F7 hex | [CompoNet Link Class] | See Chapter 7 of Volume 6, CompoNet Adaptation of CIP | |
| CompoNet Repeater Object<br>F8 hex | [CompoNet Repeater Class] | See Chapter 7 of Volume 6, CompoNet Adaptation of CIP | |

Table Footnotes

1 Required if the device supports the Connection Configuration object, not allowed if the device does not support the Connection Configuration object.

2 Required if the device supports one or more Port object instances that support CIP routing, optional otherwise.

### 7-3.6.1.2 Vendor Specific Object Class Sections

Optional Vendor Specific Object Class Sections shall be named **ClassN_VendorM** where N is the decimal vendor specific class code and M is the decimal Vendor ID. For instance, **[Class100_Vendor1534]** is the name of the class object section for class code 100, vendor 1534.

The Vendor Specific Object Class Sections may contain any of the Common Object Class Entry Keyword(s) described in section 7-3.6.1.3.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.1.3 Common Object Class Entry Keywords

Every CIP object has certain characteristics that are described by the following common Object Class Section entry keywords. These keywords may be used in any Public Object Class Section (see chapter 7-3.6.1.1) or Vendor Specific Object Class Section (see section 7-3.6.1.2).

**Table 7-3.5 Common Object Class Entry Keywords**

| Entry Name | Entry Keyword | Required/Optional |
|---|---|---|
| Revision | Revision | Required |
| Maximum Instance Number | MaxInst | Required |
| Number of Static Instances | Number_Of_Static_Instances | Required |
| Maximum Number of Dynamic Instances | Max_Number_Of_Dynamic_Instances | Required |
| Class attribute identification | Class_Attributes | Optional |
| Instance attribute identification | Instance_Attributes | Optional |
| Class service support | Class_Services | Optional |
| Instance service support | Instance_Services | Optional |
| Object Name | Object_Name | Required |
| Object Class Code | Object_Class_Code | Required |
| Service Description | Service_DescriptionN | Optional |

##### 7-3.6.1.3.1 Revision Entry Keyword

The “Revision” entry keyword corresponds to the class attribute #1 (Revision). The “Revision” entry keyword shall contain the field shown in Table 7-3.6.

**Table 7-3.6 Revision Entry Keyword Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Revision | 1 | UINT | Required |

###### 7-3.6.1.3.1.1 Revision, Field 1

This entry field contains the value that shall match the value of the object’s Revision class attribute (attribute #1).

##### 7-3.6.1.3.2 MaxInst Entry Keyword

The “MaxInst” entry keyword identifies the maximum instance number of the class. The “MaxInst” entry keyword shall contain the field shown in Table 7-3.7.

**Table 7-3.7 MaxInst Entry Keyword Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| MaxInst | 1 | UINT | Required |

###### 7-3.6.1.3.2.1 MaxInst, Field 1

This entry field contains the value that identifies the maximum (numerically largest) instance number of the class.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.1.3.3 Number_Of_Static_Instances Entry Keyword

The "Number_Of_Static_Instances" entry keyword identifies the total number of static instances of the object. The "Number_Of_Static_Instances" entry keyword shall contain the field shown in Table 7-3.8.

**Table 7-3.8 Number_Of_Static_Instances Entry Keyword Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Number of static instances | 1 | UINT | Required |

##### 7-3.6.1.3.3.1 Number_Of_Static_Instances, Field 1

This entry field contains the value that identifies the total number of static instances of the object in the device.

#### 7-3.6.1.3.4 Max_Number_Of_Dynamic_Instances Entry Keyword

The "Max_Number_Of_Dynamic_Instances" entry keyword identifies the maximum number of dynamic instances of the object capable of being created in the device. The "Max_Number_Of_Dynamic_Instances" entry keyword shall contain the field shown in Table 7-3.9.

**Table 7-3.9 Max_Number_Of_Dynamic_Instances Entry Keyword Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Maximum number of dynamic instances | 1 | UINT | Required |

##### 7-3.6.1.3.4.1 Maximum Number_Of_Dynamic_Instances Entry Keyword, Field 1

This entry field contains the value that identifies the total number of dynamic instances of the object capable of being created in the device.

#### 7-3.6.1.3.5 Class_Attributes Entry Keyword

The "Class_Attributes" entry keyword is used to identify attributes within an object class (instance number zero). The "Class_Attributes" entry keyword shall contain the fields shown in Table 7-3.10.

**Table 7-3.10 Class_Attributes Entry Keyword Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Attribute ID 1 | 1 | UINT | Required |
| Attribute ID 2 | 2 | UINT | Conditional |
| … | | | |
| Attribute ID n | N | UINT | Conditional |

##### 7-3.6.1.3.5.1 Attribute ID Entry Keyword, Field 1 thru n

These entry fields shall contain values that identify attributes of the object's class in the device. Each field in the list indicates an attribute that has been implemented in the object's class. The total number of fields is variable based upon the actual number of attributes implemented. No empty fields are allowed.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.1.3.6 Instance_Attributes Entry Keyword

The "Instance_Attributes" entry keyword is used to identify attributes within a class instance (non-zero instance number). The "Instance_Attributes" entry keyword shall contain the fields shown in Table 7-3.11.

**Table 7-3.11 Instance_Attributes Entry Keyword Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Attribute ID 1 | 1 | UINT | Required |
| Attribute ID 2 | 2 | UINT | Conditional |
| … | | | |
| Attribute ID n | N | UINT | Conditional |

##### 7-3.6.1.3.6.1 Attribute ID Entry Keyword, Field 1 thru n

These entry fields shall contain values that identify attributes of the object's instance(s) in the device. Each field in the list indicates an attribute that has been implemented in the object's instance(s). The total number of fields is variable based upon the actual number of attributes implemented. No empty fields are allowed.

#### 7-3.6.1.3.7 Class_Services Entry Keyword

The "Class_Services" entry keyword is used to identify services within an object class (instance number zero). The "Class_Services" entry keyword shall contain the fields shown in Table 7-3.12.

**Table 7-3.12 Class_Services Entry Keyword Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Service 1 | 1 | EDS_Service | Required |
| Service 2 | 2 | EDS_Service | Conditional [1] |
| … | | | |
| Service n | N | EDS_Service | Conditional [1] |

Table Footnotes

1 Not allowed if additional class services are not supported, optional otherwise.

##### 7-3.6.1.3.7.1 Class Services Entry Keyword, Field 1 thru n

These entry fields shall contain values that identify services of the object's class in the device. Each field in the list indicates a service that has been implemented in the object's class. The total number of fields is variable based upon the number of services implemented. No empty fields are allowed.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.1.3.8 Instance_Services Entry Keyword

The "Instance_Services" entry keyword is used to identify services within a class instance (non-zero instance number). The "Instance Services" entry keyword shall contain the fields shown in Table 7-3.13.

**Table 7-3.13 Instance_Services Entry Keyword Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Service 1 | 1 | EDS_Service | Required |
| Service 2 | 2 | EDS_Service | Conditional [1] |
| … | | | |
| Service n | n | EDS_Service | Conditional [1] |

Table Footnotes

1 Not allowed if additional instance services are not supported, optional otherwise.

##### 7-3.6.1.3.8.1 Instance Services Entry Keyword, Field 1 thru n

These entry fields shall contain values that identify services of the object's instance(s) in the device. Each field in the list indicates a service that has been implemented in the object's instance(s). The total number of fields is variable based upon the number of services implemented. No empty fields are allowed.

#### 7-3.6.1.3.9 Object_Name Keyword

The "Object_Name" entry keyword is used to specify the name of the object. The "Object_Name" entry keyword shall contain the fields shown in Table 7-3.14.

**Table 7-3.14 Object_Name Entry Keyword Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Name | 1 | STRING | Required |

##### 7-3.6.1.3.9.1 Name – Field 1

This entry field contains the name of the object. For public objects the name shall be the name listed in Table 7-3.4, CIP Object column.

#### 7-3.6.1.3.10 Object_Class_Code Keyword

The "Object_Class_Code" entry keyword is used to specify the class code of the object. The "Object_Class_Code" entry keyword shall contain the fields shown in Table 7-3.15.

**Table 7-3.15 Object_Class_Code Entry Keyword Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Code | 1 | UDINT | Required |

##### 7-3.6.1.3.10.1 Code – Field 1

This entry field contains the class code of the object.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.1.3.11 Service_DescriptionN Entry Keywords

The Service Description keyword is defined to allow a device developer to describe services supported by this object in the device. This information can be used by a configuration tool to:

1. Provide a user interface for the user to configure the transmission of a service, including specifying the parameters in the service request data.
2. Provide a user interface for the user to view the parameters in a service response.

Both class and instance services can be described using this entry keyword. In addition, if an error response is received, the error code information can be documented in an EventN entry and a configuration tool can provide a user interface to display the error information.

For the case when a configuration tool knows the details of a service described in an EDS file, the tool can decide to use either the details it already knows or use the information from the EDS file. Get and set attribute service support is described by ParamN and AssemN entries that specify an EPath. However, get and set attribute services may be defined using the Service_DescriptionN keyword because a specific Name can be assigned that provides a better description.

The Service Description keyword shall consist of the name "**Service_Description**" combined with a number (decimal). Each entry describes one service. Multiple Service_DescriptionN entries shall not exist that define the same service request (a service request consists of a service code, application path and service request data). Multiple Service_DescriptionN entries are allowed with the same service code as long as the application path and service request data are not also identical.

The "**Service_Description**" entry keywords define the details associated with a service, including the service code, name, service application path, the format of the service request data and the format of the service response data.

Each "**Service_DescriptionN**" entry shall contain the formatted fields shown in Table 7-3.16.

**Table 7-3.16 Service_Description Entry Keyword Format**

| Field Name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Service Code | 1 | USINT | Required |
| Name | 2 | STRING | Required |
| Service Application Path | 3 | Padded EPATH | Required |
| Service Request Data | 4 | Empty, AssemExaN, ParamN | Conditional[1] |
| Service Response Data | 5 | Empty, AssemExaN, ParamN | Conditional[2] |

Table Footnotes:

1. Required if the service has request data, not allowed (empty) if the service does not have request data.
2. Required if the service has response data, not allowed (empty) if the service does not have response data.

Figure 7-3.6 below, shows an example using the Service_DescriptionN keyword.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-3.6 Sample Electronic Data Sheet Illustrating the Service_DescriptionN keyword**

```
.
.
.

[Ethernet Link Class]
   Revision = 3;
   Object_Class_Code = 0xF6;
   Object_Name = "Ethernet Link Object”;
   MaxInst = 2;
   Number_Of_Static_Instances = 2;
   Max_Number_Of_Dynamic_Instances = 0;
   InterfaceLabel1 = "1";
   InterfaceLabel2 = "2";
   InterfaceType1 = 2;           $ Twisted Pair
   InterfaceType2 = 2;           $ Twisted Pair
   Service_Description1 =        $ ENET Link Object get and clear instance
                                 $ Interface Counters description
      0x4C,                      $ Service Code
      "Get and Clear Ethernet Interface Counters",  $ Service Description
                                                    $ Name
      "20 F6 24 [Param1] 30 04", $ Application Path
      ,                          $ No service request data
      Assem1;                    $ Service response data

[Params]
   Param1 =
     0,                      $ First field shall equal 0
     ,,                      $ no path, this item is used to construct the
                             $ Service_Description1 application path
     0x0000,                 $ descriptor
     0xC6,                   $ USINT
     1,                      $ data size
     "Interface",            $ name
     "",                     $ no units
     "",                     $ no help string
     ,,1,                    $ no min or max (enumerated values), default
                             $ data value is Interface Number 1
     ,,,,                    $ no mult, div, base, offset scaling
     ,,,,                    $ no mult, div, base, offset links not used
     ;                       $ no decimal places

    Enum1 =
      1, "Port 1", 2, " Port 2";

    Param2 =
      0,                      $ First field shall equal 0
      ,,                      $ no path, this item is used to construct the
                              $ Interface Counters assembly
      0x0000,                 $ descriptor
      0xC8,                   $ UDINT
      4,                      $ data size
      "In Octets",            $ name
      "",                     $ no units
      "",                     $ no help string
      ,,0,                    $ no min or max (enumerated values), default
                              $ value is 0
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
    ,,,,                    $ no mult, div, base, offset scaling
    ,,,,                    $ no mult, div, base, offset links not used
    ;                       $ no decimal places

  Param3 =
    0,                      $ First field shall equal 0
    ,,                      $ no path, this item is used to construct the
                            $ Interface Counters assembly
    0x0000,                 $ descriptor
    0xC8,                   $ UDINT
    4,                      $ data size
    "In Ucast Packets",     $ name
    "",                     $ no units
    "",                     $ no help string
    ,,0,                    $ no min or max (enumerated values), default
                            $ value is 0
    ,,,,                    $ no mult, div, base, offset scaling
    ,,,,                    $ no mult, div, base, offset links not used
    ;                       $ no decimal places

  Param4 =
    0,                      $ First field shall equal 0
    ,,                      $ no path, this item is used to construct the
                            $ Interface Counters assembly
    0x0000,                 $ descriptor
    0xC8,                   $ UDINT
    4,                      $ data size
    "In NUcast Packets",    $ name
    "",                     $ no units
    "",                     $ no help string
    ,,0,                    $ no min or max (enumerated values), default
                            $ value is 0
    ,,,,                    $ no mult, div, base, offset scaling
    ,,,,                    $ no mult, div, base, offset links not used
    ;                       $ no decimal places

  Param5 =
    0,                      $ First field shall equal 0
    ,,                      $ no path, this item is used to construct the
                            $ Interface Counters assembly
    0x0000,                 $ descriptor
    0xC8,                   $ UDINT
    4,                      $ data size
    "In Discards",          $ name
    "",                     $ no units
    "",                     $ no help string
    ,,0,                    $ no min or max (enumerated values), default
                            $ value is 0
    ,,,,                    $ no mult, div, base, offset scaling
    ,,,,                    $ no mult, div, base, offset links not used
    ;                       $ no decimal places

  Param6 =
    0,                      $ First field shall equal 0
    ,,                      $ no path, this item is used to construct the
                            $ Interface Counters assembly
    0x0000,                 $ descriptor
    0xC8,                   $ UDINT
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
    4,                      $ data size
    "In Errors",            $ name
    "",                     $ no units
    "",                     $ no help string
    ,,0,                    $ no min or max (enumerated values), default
                            $ value is 0
    ,,,,                    $ no mult, div, base, offset scaling
    ,,,,                    $ no mult, div, base, offset links not used
    ;                       $ no decimal places

  Param7 =
    0,                      $ First field shall equal 0
    ,,                      $ no path, this item is used to construct the
                            $ Interface Counters assembly
    0x0000,                 $ descriptor
    0xC8,                   $ UDINT
    4,                      $ data size
    "In Unknown Protos",    $ name
    "",                     $ no units
    "",                     $ no help string
    ,,0,                    $ no min or max (enumerated values), default
                            $ value is 0
    ,,,,                    $ no mult, div, base, offset scaling
    ,,,,                    $ no mult, div, base, offset links not used
    ;                       $ no decimal places

  Param8 =
    0,                      $ First field shall equal 0
    ,,                      $ no path, this item is used to construct the
                            $ Interface Counters assembly
    0x0000,                 $ descriptor
    0xC8,                   $ UDINT
    4,                      $ data size
    "Out Octets",           $ name
    "",                     $ no units
    "",                     $ no help string
    ,,0,                    $ no min or max (enumerated values), default
                            $ value is 0
    ,,,,                    $ no mult, div, base, offset scaling
    ,,,,                    $ no mult, div, base, offset links not used
    ;                       $ no decimal places

  Param9 =
    0,                      $ First field shall equal 0
    ,,                      $ no path, this item is used to construct the
                            $ Interface Counters assembly
    0x0000,                 $ descriptor
    0xC8,                   $ UDINT
    4,                      $ data size
    "Out Ucast Packets",    $ name
    "",                     $ no units
    "",                     $ no help string
    ,,0,                    $ no min or max (enumerated values), default
                            $ value is 0
    ,,,,                    $ no mult, div, base, offset scaling
    ,,,,                    $ no mult, div, base, offset links not used
    ;                       $ no decimal places
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
    Param10 =
      0,                      $ First field shall equal 0
      ,,                      $ no path, this item is used to construct the
                              $ Interface Counters assembly
      0x0000,                 $ descriptor
      0xC8,                   $ UDINT
      4,                      $ data size
      "Out NUcast Packets",   $ name
      "",                     $ no units
      "",                     $ no help string
      ,,0,                    $ no min or max (enumerated values), default
                              $ value is 0
      ,,,,                    $ no mult, div, base, offset scaling
      ,,,,                    $ no mult, div, base, offset links not used
      ;                       $ no decimal places

    Param11 =
      0,                      $ First field shall equal 0
      ,,                      $ no path, this item is used to construct the
                              $ Interface Counters assembly
      0x0000,                 $ descriptor
      0xC8,                   $ UDINT
      4,                      $ data size
      "Out Discards",         $ name
      "",                     $ no units
      "",                     $ no help string
      ,,0,                    $ no min or max (enumerated values), default
                              $ value is 0
      ,,,,                    $ no mult, div, base, offset scaling
      ,,,,                    $ no mult, div, base, offset links not used
      ;                       $ no decimal places

    Param12 =
      0,                      $ First field shall equal 0
      ,,                      $ no path, this item is used to construct the
                              $ Interface Counters assembly
      0x0000,                 $ descriptor
      0xC8,                   $ UDINT
      4,                      $ data size
      "Out Errors",           $ name
      "",                     $ no units
      "",                     $ no help string
      ,,0,                    $ no min or max (enumerated values), default
                              $ value is 0
      ,,,,                    $ no mult, div, base, offset scaling
      ,,,,                    $ no mult, div, base, offset links not used
      ;                       $ no decimal places

[Assembly]
    Assem1 =
      "Interface Counters",   $ Name
      ,                       $ No Path
      ,                       $ No Size
      0x0000,                 $ descriptor
      ,,                      $ Reserved, Reserved
      ,Param2,                $ In Octets
      ,Param3,                $ In Ucast Packets
      ,Param4,                $ In NUcast Packets
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
      ,Param5,                $ In Discards
      ,Param6,                $ In Errors
      ,Param7,                $ In Unknown Protos
      ,Param8,                $ Out Octets
      ,Param9,                $ Out Ucast Packets
      ,Param10,               $ Out NUcast Packets
      ,Param11,               $ Out Discards
      ,Param12;               $ Out Errors

.
.
.
```

#### 7-3.6.1.3.11.1 Service Code, Field 1

The first field, called "Service Code", is the service code associated with the service, used as the Message Router Request Service parameter. See object descriptions for the definition of services supported by the object.

#### 7-3.6.1.3.11.2 Name, Field 2

The second field, called "Name", is the name to be displayed to a user. This should be a descriptive name that will help the user when they are making a selection to understand the action this will trigger. The contents of the "Name" field shall be unique amongst all Service_DescriptionN keywords.

#### 7-3.6.1.3.11.3 Service Application Path, Field 3

The third field, called "Service Application Path", is the application path used as the Message Router Request Request_Path parameter (see section 2-4.1). Notice that the application path must include the Class ID encoding. Class services will specify an Instance ID of zero. The Class ID encoded in the application path must match the object code for object defined by this section.

NOTE: A service description can describe a service to a single application path or to a set of application paths. For a service description to describe a set of application paths, one or more [ParamN] entries can be included in the path. For example "20 64 24 [Param1] 30 [Param2]" would describe a service that is sent to one of a number of object instances and also to one of a number of attributes that would identified in the respective ParamN entries.

#### 7-3.6.1.3.11.4 Service Request Data, Field 4

The fourth field, called "Service Request Data", is the service request data used as the Message Router Request Request_Data parameter (see section 2-4.1).

#### 7-3.6.1.3.11.5 Service Response Data, Field 5

The fifth field, called "Service Response Data", is the service request data received in the Message Router Response Response Data parameter (see section 2-4.2) when a successful response is received. If an error response is received, the error code information can be documented in an EventN entry.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.6.2 File Description Section

The file description section begins with the keyword [File] and contains administrative information about the EDS file. A configuration tool reads this information, formats it, and displays it to the user. The user can also access this section with a text file viewer and display the unformatted information. This section requires no modification unless the user manually modifies the file. The file description section must contain:

**Table 7-3.17 Keywords in the File Description Section**

| Entry Name | Entry Keyword | Field Number | Data Type | Required/Optional |
|---|---|---|---|---|
| File Description Text | DescText | 1 | STRING | Required |
| File Creation Date | CreateDate | 1 | EDS_Date | Required |
| File Creation Time | CreateTime | 1 | EDS_Time_of_Day | Required |
| Last Modification Date | ModDate | 1 | EDS_Date | Conditional |
| Last Modification Time | ModTime | 1 | EDS_Time_of_Day | Conditional |
| EDS Revision | Revision | 1 | EDS_Revision | Required |
| Home URL | HomeURL | 1 | EDS_URL | Optional |
| Exclude | Exclude | 1 | EDS_Field_Keyword allowed values: NONE, WRITE, READ_WRITE | Optional |
| EDS File CRC | EDSFileCRC | See CIP Safety Specification (Volume 5, Chapter 7) | | Conditional 1 |

Table Footnotes:

1. This keyword is not allowed if the device is not a safety device. If the device is a safety device, see Volume 5.

The entries in the file description section provide:

- **File Description Text -** a single line of text displayed by the configuration tool. The EDS developer assigns a meaningful line of text for this entry. Double quotes enclose all character arrays.
- **File Creation Date -** the creation date of the EDS, assigned by the EDS developer. Provided only for convenience, you can use this date to get version information about the file. A configuration tool does not use this information to perform any type of version control, but it may display the contents.
- **File Creation Time -** the creation time of the EDS, assigned by the EDS developer. Provided only for convenience, you can use this date to get version information about the file. A configuration tool does not use this information to perform any type of version control, but it may display the contents.
- **Last Modification Date -** the date of the last modification to the EDS. A configuration tool that allows modification of the EDS file shall update this field as needed. Provided only for convenience, the configuration tool displays the contents of this entry if it exists. If a configuration tool changes the EDS, the configuration tool shall update this field. However, if you modify the EDS manually or with a text editor, you should also update this field.

  This keyword is required if either:

  - the EDS file is modified by a software tool, or
  - the Last Modification Time keyword is present


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- **Last Modification Time -** the time of the last modification to the EDS. A configuration tool that allows modification of the EDS file must update this entry as needed. Provided for convenience, the configuration tool displays the contents of this entry if it exists. If a configuration tool changes the EDS, the configuration tool must update this field. However, if you modify the EDS manually or with a text editor, you should also update this field.
- **EDS Revision -** the revision of the EDS. The EDS revision is not required to have any relationship to the product's revision, it is simply the revision of the EDS file itself.

  When comparing the EDS Revision field value between EDS files for the same device (same device means the field values for the VendCode, ProdType, ProdCode and MajRev keywords in the Device section in both EDS files match) the following algorithm shall be used to determine which EDS file is most recent:

  EDS 1 Revision is major_revision1.minor_revision1

  EDS 2 Revision is major_revision2.minor_revision2

  if ((major_revision1 > major_revision2) OR ((major_revision1 EQUALS major_revision2) AND (minor_revision1 > minor_revision2)))

  EDS 1 is most recent

  Else if ((major_revision1 < major_revision2) OR ((major_revision1 EQUALS major_revision2) AND (minor_revision1 < minor_revision2)))

  EDS 2 is most recent

  Else

  EDS 2 and EDS 1 are the same revision
- **Home URL –** Uniform Resource Locator of the master EDS file, the Icon file and other files related to this EDS. The Home URL shall specify a complete qualified URL for referencing a master version of the EDS file. In addition, the referenced area (without the file name specification) is used to specify an area where other related file(s) relating to the device described by this EDS are contained.
- **Exclude –** This entry keyword, when present, indicates that the manufacturer of the device expects a specific behavior by the configuration tool if that tool encounters any items (section keywords, entry keywords, field values, etc), which are not understood. The behavior modification of the tool is defined as which type of operations shall be excluded when any items, which are not understood are encountered.

  The meaning of the allowed EDS_Field_Keyword values shall be interpreted as:

  - **NONE –** No behaviors are excluded, this is the default if the Exclude entry keyword does not exist.
  - **WRITE –** All write access to the device shall be excluded (e.g. no Set_Attribute_Single, Set_Member, etc. service shall be performed). The configuration tool shall also exclude configuring a second device, which results in the second device writing to the device described by this EDS.
  - **READ_WRITE –** All access to the device is excluded (e.g. no Get_Attribute_Single, Get_Member, etc. service shall be performed). The configuration tool shall also exclude configuring a second device, which results in the second reading from or writing to the device described by this EDS.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

When a software tool prevents writing or reading/writing a device due to the Exclude keyword messages shall be displayed by the software tool indicating why the operation is excluded. The following text strings shall be used:

- When excluding writing a device because the software tool does not understand a vendor specific item (section keywords, entry keywords, field values, etc) – "This device cannot be configured because the device supports some vendor-specific behavior that is not supported by this software."
- When excluding writing a device because the software tool does not understand an open item (section keywords, entry keywords, field values, etc) – "This device cannot be configured because this software does not support a CIP defined entry described in this devices' EDS file. Contact the vendor of this software for an updated version."
- When excluding reading/writing a device because the tool does not understand a vendor specific item (section keywords, entry keywords, field values, etc.) – "The configuration for this device cannot be displayed because the device supports some vendor-specific behavior that is not supported by this software."
- When excluding reading/writing a device because the tool does not understand an open item (section keywords, entry keywords, field values, etc.) – "The configuration for this device cannot be displayed because this software does not support a CIP defined entry described in this devices' EDS file. Contact this vendor of the software for an updated version."
- When multiple conditions apply, (both vendor specific and open items are not understood) the tool shall display both text strings.

**Figure 7-3.7 Sample Electronic Data Sheet Illustrating the File Section**

```
[File]
   DescText = "Smart Widget EDS File";
   CreateDate = 04-03-1994;       $ created
   CreateTime = 17:51:44;
   ModDate = 04-06-1994;          $ last changed
   ModTime = 22:07:30;
   Revision = 2.1;                $ Revision of EDS
   HomeURL = "http://www.odva.org/EDS/example.eds";

[Device]
   ...
[ParamClass]
   ...
[Params]
   ...
[Groups]
   ...
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.6.3 Device Description Section

The Device Description section begins with the keyword [Device] and contains a manufacturer's information about the device, including some of the same values in a device's Identity Object. The table below shows the entries in the Device Description section.

**Table 7-3.18 Keywords in the Device Description Section**

| Entry Name | Entry Keyword[1] | Field Number | Data Type | Required/ Optional |
|---|---|---|---|---|
| Vendor ID[2,3] | VendCode | 1 | UINT | Required |
| Vendor Name | VendName | 1 | STRING | Required |
| Device Type[2,3] | ProdType | 1 | UINT | Required |
| Device Type String | ProdTypeStr | 1 | STRING | Required |
| Product Code[2,3] | ProdCode | 1 | UINT | Required |
| Major Revision[2,3] | MajRev | 1 | USINT | Required |
| Minor Revision[2] | MinRev | 1 | USINT | Required |
| Product Name[2] | ProdName | 1 | STRING | Required |
| Catalog Number | Catalog | 1 | STRING | Optional |
| Exclude from Adapter Rack Connection | ExcludeFromAdapterRackConnection | 1 | Field Keyword, possible values: Yes, No | Optional |
| Icon File Name | Icon | 1 | STRING | Conditional [4] |
| Icon Contents | IconContents | 1 | STRING | Optional |
| Vendor-specific Extended Device Status Assembly | DeviceStatusAssembly | 1 | UDINT | Conditional [5] |
| Vendor-specific Extended Device Status Assembly 2 | DeviceStatusAssembly 2 | 1 | UDINT | Conditional [6] |

Table Footnotes

1 The SerNum and Comment entry keywords have been obsoleted

2 This entry represents an attribute of the Identity Object

3 This entry is used to match an EDS with the values in Instance 1 of the Identity Object

4 Required if IconContents is present in this section, optional otherwise.

5 Required if device implements a Vendor-specific Extended Device Status field definition, not allowed otherwise.

6 Required if device implements a Vendor-specific Extended Device Status 2 field definition, not allowed otherwise.

The entry name for the Device Description field describes the unique data entry line number. A configuration tool uses the Vendor ID, Device Type, Product Code and Major Revision entries in the Device Description section to match the EDS to the device being configured. The entries in the device description section provide:

- **Vendor ID -** Numeric vendor identifier as defined by the Identity Object, Instance 1, Attribute 1.
- **Vendor Name -** Textual vendor name. When displayed, truncation may occur to meet the display capabilities.
- **Device Type -** Numeric device identifier as defined by the Identity Object, Instance 1, Attribute 2.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- **Device Type String -** Textual description of device type exactly as shown in Chapter 6-7; the only exceptions are the parenthetical comments behind the two Generic Device Types, these shall be omitted. The string for vendor specific device types is at the vendors' discretion.
- **Product Code -** Vendor assigned numeric product code identifier as defined by the Identity Object, Instance 1, Attribute 3. Each product code shall have its own EDS.
- **Major Revision -** Vendor-assigned major revision number as defined by the Identity Object, Instance 1, Attribute 4. The major revision of a product is typically incremented when there is a change to the form, fit, or function of the device. Changes to major revisions are used by a configuration tool to match a device to an EDS.
- **Minor Revision -** Vendor-assigned minor revision number as defined by the Identity Object, Instance 1, Attribute 4. The minor revision number is used to identify changes in a product that do not affect user configuration choices. For example: firmware bug fixes, an additional LED, internal hardware changes, etc. Changes in minor revisions are not used by a configuration tool to match a device with an EDS. When online with the device and displaying Minor Revision, a tool should display the value read from the device instead of the one read from the EDS.
- **Product Name -** Textual product name as defined by the Identity Object, Instance 1, Attribute 7. When displayed, truncation may occur to meet the display capabilities. The Product Name string in the EDS may differ from the string found in the Identity Object, Instance 1, Attribute 7 since an EDS may cover product variants that present the same logical view to the network. When online with the device and displaying Product Name, a tool should display the Product Name read from the device instead of the one read from the EDS.
- **Catalog Number -** Textual catalog or model number. One or more catalog numbers may be associated with a particular product code. In the case of multiple catalog numbers, it is still useful to provide as much of the catalog number as is practical. For example, 1438-BAC7xx where 'xx' represents variants in the catalog number supported by this product code/EDS.
- **ExcludeFromAdapterRackConnection -** This field is used to describe if a rack-based device must be excluded from an adapter rack connection. If the field value is the EDS_Field_Keyword "Yes" this module shall be excluded from adapter rack connections by resetting the associated slot mask bits (input, output and configuration). If the field value is the EDS_Field_Keyword "No" or this optional field is omitted the associated slot mask bits may be set.
- **Icon File Name-** File name of an Icon file. Identifies a file that contains a graphical representation of the device. The file shall have the *.ICO MSWindows format, and shall minimally contain a 16x16 icon. The file may also contain 32x32, 48x48, and 64x64 icons. If the HomeURL keyword exists, the location of the Icon file is the combination of the location specified by the HomeURL keyword (without the HomeURL file name component) and the file name specified by this keyword. This keyword shall only be present when a HomeURL keyword or IconContents keyword exists. Further, this keyword shall be present if the IconContents keyword is present.

  A vendor shall not specify the same Icon File Name for more than one icon. Thus, the Icon File Name can be used by a configuration tool to identify when more than one EDS file from a vendor uses the same icon.
- **Icon Contents –** ASCII encoded contents of the Icon file for the device.

  The encoding algorithm is one of several Base64 algorithms in use. The specific algorithm used is as follows:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

To encode the data, the first byte is placed in the most significant eight bits of a 24-bit buffer, the next byte in the middle eight bits, and the third byte in the least significant eight bits. The buffer is then used, six bits at a time, most significant first, as indices into the string:

"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/", and the indicated character is output.

The process is repeated on the remaining data until less than three bytes remain. If fewer than three bytes (24 bits) are remaining to encode, the input data is padded with zero bits to form an integral multiple of six bits. After encoding padded data, if two bytes were remaining to encode, one "=" character is appended to the output; if one byte was remaining, two "=" characters are appended. This signals the decoder that there were less than 3 bytes of data encoded and there were pad bits appended during encoded.

Example:

Unencoded hexadecimal: “00 00 00” Encoded: “AAAA”

Unencoded hexadecimal: “FF FF FF” Encoded: “////”

Unencoded hexadecimal: “00 00 00 00” Encoded: “AAAAAA==”

Unencoded hexadecimal: “00 00 00 00 00” Encoded: “AAAAAAA=”

- **Vendor-specific Extended Device Status Assembly –** Identifies the value (N) of the AssemN or AssemExaN that defines the vendor-specific format of the Extended Device Status field (bits 4-7) of the Identity Object Status instance attribute (see section 5A-2.2.1.5 Status).
- **Vendor-specific Extended Device Status Assembly 2 –** Identifies the value (N) of the AssemN or AssemExaN that defines the vendor-specific format of the Extended Device Status 2 field (bits 12-15) of the Identity Object Status instance attribute (see section 5A-2.2.1.5 Status).

**Figure 7-3.8 Sample Electronic Data Sheet Illustrating the Device Description Section**

```
[File]
   ...
[Device]
   VendCode = 65535;
   VendName = "Widget-Works, Inc.";
   ProdType = 43;
   ProdTypeStr = “Generic Device”;
   ProdCode = 42;
   MajRev = 1;                    $ Device Major Revision
   MinRev = 1;                    $ Device Minor Revision
   ProdName = "Smart-Widget";
   Catalog = “1499-DVG”;
   Icon = “example.ico”;
   DeviceStatusAssembly = 1;
   DeviceStatusAssembly2 = 2;
[ParamClass]
   ...
[Assembly]
   Assem1 =
      “Device Status, bits 4-7”, $ name
      ,                           $ no path
      ,                           $ size - unspecified
      ,                           $ reserved
      ,,                          $ reserved
      4, Param1;                  $ describes bits 4-7
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
   Assem2 =
      “Device Status, bits 12-15”, $ name
      ,                            $ no path
      ,                            $ size - unspecified
      ,                            $ reserved
      ,,                           $ reserved
      4, Param2;                   $ describes bits 12-15
   ...
[Params]
    Param1 =
      0,                  $ reserved, shall equal 0
      ,,                  $ Link Path Size, Link Path
      0x0030,             $ Descriptor
      0xC6,               $ Data Type
      1,                  $ Data Size in bytes
      "Device Status Values, bits 4-7", $ name
      "",                 $ units
      "",                 $ help string
      ,,0,                $ min, max, default values
      ,,,,                $ mult, div, base, offset scaling
      ,,,,                $ mult, div, base, offset links
      ;                   $ decimal places
    Enum1 =  $ values 0-7 reflect the open definitions in table 5A-2.4
combined with vendor specific values 10 - 13
     0,"Self-Testing or Unknown",
     1,"Firmware Update in Progress",
     2,"At least one faulted I/O connection",
     3,"No I/O connections established",
     4,"Non-Volatile Configuration bad",
     5,"Major Fault – either bit 10 or bit 11 is true (1)",
     6,"At least one I/O connection in run mode",
     7,"At least one I/O connection established, all in idle mode",
     10,"Stored Configuration/Program Invalid",
     11,"Backplane not found",
     12,"All I/O connections in run mode",
     13,"All I/O connections in idle mode";

    Param2 =
      0,                  $ reserved, shall equal 0
      ,,                  $ Link Path Size, Link Path
      0x0030,             $ Descriptor
      0xC6,               $ Data Type
      1,                  $ Data Size in bytes
      "Device Status Values, bits 12-15", $ name
      "",                 $ units
      "",                 $ help string
      ,,0,                $ min, max, default values
      ,,,,                $ mult, div, base, offset scaling
      ,,,,                $ mult, div, base, offset links
      ;                   $ decimal places
    Enum2 =  $ values 0-3 reflect vendor specific definitions
      0,"Unknown – not supported",
      1,"Mode Transitioning",
      2,"Test Mode",
      3,"Keyswitch in remote mode";   ...
[Groups]
   ...
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.6.4 Device Classification Section

The Device Classification section begins with the keyword [Device Classification] and shall classify the device described by the EDS into one or more categories of devices. The entry keyword for all classifications shall consist of the character array, "Class", combined with a decimal number. The numbers shall start at 1 for the first class, and shall be incremented for each additional class. Commas shall separate all fields, and a semicolon shall indicate the end of the entry.

The fields for each classification entry are shown in Table 7-3.19

**Table 7-3.19 Fields in the ClassN Keyword**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Classification 1 | 1 | EDS_Field_Keyword, allowed values: CompoNet, ControlNet, DeviceNet, EtherNetIP, ModbusSL, ModbusTCP, Safety | Required |
| Classification N | 2, 3, 4, … | EDS_Field_Keyword, allowed values: CompoNet, ControlNet, DeviceNet, EtherNetIP, ModbusSL, ModbusTCP, Safety | Conditional [1] |

1 Not allowed if Classification 1 is a public classification, optional if Classification 1 is a vendor specific classification.

#### 7-3.6.4.1 Classification Field

The classification field value can be either a public classification or a vendor specific classification. The public classifications are:

- CompoNet
- ControlNet
- DeviceNet
- EtherNetIP
- ModbusSL
- ModbusTCP
- Safety

The first classification field is required. When the first classification field is a public value, no other classification fields are allowed.

The vendor-specific field shall begin with the Vendor ID of the company making the addition followed by an underscore (*VendorID_VendorSpecificField)*. The VendorID shall be displayed in decimal and shall not contain leading zeroes. Each vendor is responsible for maintaining and documenting their vendor-specific field.

### 7-3.6.5 Parameter Class Section

The parameter class section begins with the keyword [ParamClass]. The purpose of this section is to:

- identify the class level attributes of the configuration parameters
- contain a subset of the Parameter Object class attributes as defined in the CIP Object Library


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The following table identifies the entry keywords in the Parameter Class section that differ from the Common Object Class definition:

**Table 7-3.20 Parameter Class Section Entry Keyword Changes/Additions to the Common Object Class Entry Keywords (Table 7-3.5)**

| Entry Name | Entry Keyword | Required/Optional |
|---|---|---|
| Revision | Revision | Optional [2] |
| Number of Static Instances | Number_Of_Static_Instances | Optional [2] |
| Maximum Number of Dynamic Instances | Max_Number_Of_Dynamic_Instances | Optional [2] |
| Parameter Class Descriptor | Descriptor | Required |
| Configuration Assembly Instance | CfgAssembly | Required |
| Safety Assembly Instance | SafetyCfgAssembly | Conditional [1] |
| Configuration Assembly Extension A | CfgAssemblyExa | Optional [3] |

1 This keyword is not allowed if the device is not a safety device. If the device is a safety device, see Volume 5.

2 This keyword, although defined in section 7-3.6.1.3, Common Object Class Entry Keywords as Required, is Optional in the [ParamClass] section.

3 The CfgAssemblyExa keyword is only allowed if the CfgAssembly keyword has a value of 0.

### 7-3.6.5.1 Descriptor Entry Keyword

Contains bit flags that describe the behavior of the device's parameter objects.

**Table 7-3.21 Descriptor Entry Keyword Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Flags | 1 | WORD | Required |

### 7-3.6.5.2 CfgAssembly Entry Keyword

Identifies the configuration assembly object instance number.

**Table 7-3.22 CfgAssembly Entry Keyword Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Instance [1] | 1 | UINT | Required |

1 If the device does not support a configuration assembly object instance the field value shall be 0. If the value is non-zero, the structure of the configuration assembly shall be defined using an AssemN, ProxyAssemN, ProxyAssemExaN or AssemExaN keyword, whose path field specifies the configuration assembly

### 7-3.6.5.3 SafetyCfgAssembly Entry Keyword

This entry keyword is a safety-specific keyword. Use of this entry keyword is described in the CIP Safety Volume.

### 7-3.6.5.4 Configuration Assembly Extension A

This entry keyword is used when the device does not support a configuration assembly object instance, but supports a set of sub-assembly/parameter entries that are used to configure the device (e.g. the entries that should be set when the device is replaced).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The AssemN, ProxyAssemN, AssemExaN, ProxyAssemExaN entry may exclude an EPath for the case when the device supports a set of sub-assembly/parameter entries for configuration. When the AssemN, ProxyAssemN, AssemExaN, ProxyAssemExaN entry does not specify an EPath, all the members must ultimately resolve to an EPath.

**Table 7-3.23 CfgAssemblyExa Keyword Format**

| Field name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Specification | 1 | AssemN, ProxyAssemN, AssemExaN, ProxyAssemExaN, ArrayN | Required |

Figure 7-3.9 below, shows an example using the CfgAssemblyExa keyword.

**Figure 7-3.9 CfgAssemblyExa Example**

```
[ParamClass]
  CfgAssembly = 0;         $ must be 0 to use CfgAssemblyExa keyword
  CfgAssemblyExa = Assem1; $ the EDS assembly that lists the
                           $ configuration items for this device
[Params]
  Param1 =
           0,                      $ reserved, shall equal 0
           6,"20 0F 24 01 30 01",  $ Path Size, Path
           0x0000,                 $ Descriptor
           0xC6,                   $ Data Type - USINT
           1,                      $ Data Size in bytes
           "Temperature format",   $ name
           "",                     $ no units
           “Select temperature format for the device", $ help string
           0,2,0,                  $ min, max, default data values
           ,,,,                    $ mult, div, base, offset scaling
           ,,,,                    $ mult, div, base, offset links
           ;                       $ decimal places
  Enum1 =
           0, “Fahrenheit:, 1, “Celsius”, 2, “Kelvin”;
  Param2 =
           0,                      $ reserved, shall equal 0
           6,"20 68 24 01 30 01",  $ Path Size, Path
           0x0000,                 $ Descriptor
           0xC7,                   $ Data Type - UINT
           2,                      $ Data Size in bytes
           "Desired Device Temperature", $ name
           "",                     $ no units
           “Select desired device temperature", $ help string
           ,,0,                    $ min, max, default data values
           ,,,,                    $ mult, div, base, offset scaling
           ,,,,                    $ mult, div, base, offset links
           ;                       $ decimal places

  Param48 =
           0,                      $ reserved, shall equal 0
           6,"20 0F 24 30 30 01",  $ Path Size, Path
           0x0000,                 $ Descriptor
           0xC7,                   $ Data Type - UINT
           2,                      $ Data Size in bytes
           "Actual Device Temperature", $ name
           ,                       $ units
           “Device internal temperature",  $ help string
           ,,0x0055,               $ min, max, default data values
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
           ,,,,                    $ mult, div, base, offset scaling
           ,,,,                    $ mult, div, base, offset links
           ;                       $ decimal places
[Assembly]
   Assem1 =
        "Configuration",  $ name
        ,                 $ No path, the individual members specify a path
        ,                 $ Size - unspecified
        ,                 $ Descriptor
        ,,                $ Reserved
        ,Param1,          $ Member size, Member reference
        ,Param2;          $ Member size, Member reference
```

**Figure 7-3.10 Sample Electronic Data Sheet Illustrating the Parameter Class Section**

```
[File]
   ...
[Device]
   ...
[ParamClass]
   Revision = 1;
   MaxInst = 3;
   Number_Of_Static_Instances = 3;
   Max_Number_Of_Dynamic_Instances = 0;
   Descriptor = 0x0E;
   CfgAssembly = 3;

[Params]
   ...

[Groups]
   ...
```

### 7-3.6.6 Parameters Section

The parameters section begins with the keyword [Params] and identifies all of the configuration parameters in a device. All configuration performed on a device depends on the configuration parameters.

The CIP Parameter Object definition provides more specific details about the parameter fields, including the allowed data types and lengths. In general, the EDS parameter fields follow the definitions of the CIP Parameter Object instance attributes, except as noted below.

The parameter section shall identify the configuration parameters in a device.

The following is a table of the keywords that can be used in the Parameter Section:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 7-3.24 Parameter Section Entry Keywords**

| Entry Name | Entry Keyword | Required/Optional |
|---|---|---|
| Parameter | ParamN | Optional |
| Proxy Parameter | ProxyParamN | Optional |
| Proxied Parameter | ProxiedParamN | Optional |
| Enumeration | EnumN | Conditional[1] |
| Proxy Enumeration | ProxyEnumN | Conditional[1] |
| Proxied Enumeration | ProxiedEnumN | Conditional[1] |
| Fixed | FixedN | Conditional[1] |
| Constructed Parameter | ConstructedParamN | Optional |
| Proxy Constructed Parameter | ProxyConstructedParamN | Optional |
| Proxied Constructed Parameter | ProxiedConstructedParamN | Optional |

Table Footnotes

1. A corresponding ParamN, ProxyParamN, or ProxiedParamN keyword shall exist.

### 7-3.6.6.1 Param Keyword

The entry keyword shall be one of the following character arrays, "**Param**", "**ProxyParam**", "**ProxiedParam**", combined with the Parameter object instance number (decimal) for the device, e.g. "**Param1**". When a parameter object instance exists within a node, and if this parameter is also described within an EDS, then the value of "N" in "**ParamN**" shall be equal to the parameter object instance if the Link Path specified is the null string (""). When the "ParamN" does not exist within a node then the allowed number range of N shall be identical to the UDINT data type, except that 0 is not allowed. The actual parameter object instance may, but need not be, implemented in the device. Conversely, it is not required that ALL parameter object instances have a corresponding "**ParamN**" entry in an EDS. Commas shall separate all fields, and a semicolon shall indicate the end of an entry. Each entry contains the formatted fields shown in the table below. The "**ProxyParam**"and "**ProxiedParam**" keywords are defined further in Chapter 7-3.7, Modular EDS File Requirements.

**Table 7-3.25 ParamN Entry Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Reserved | 1 | USINT | Required |
| Link Path Size | 2 | USINT | Conditional |
| Link Path | 3 | Packed EPATH | Conditional |
| Descriptor | 4 | WORD | Required |
| Data Type | 5 | USINT [3] | Required |
| Data Size | 6 | USINT | Required |
| Parameter Name | 7 | STRING | Required |
| Units String | 8 | STRING | Required |
| Help String | 9 | STRING | Required |
| Minimum Value | 10 | See footnote 4 | Conditional [1] |
| Maximum Value | 11 | See footnote 4 | Conditional [1] |
| Default Value | 12 | See footnote 5 | Required |
| Scaling Multiplier | 13 | UINT | Optional |
| Scaling Divider | 14 | UINT | Optional |
| Scaling Base | 15 | UINT | Optional |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Scaling Offset | 16 | DINT[2] | Optional |
| Multiplier Link | 17 | UINT | Optional |
| Divisor Link | 18 | UINT | Optional |
| Base Link | 19 | UINT | Optional |
| Offset Link | 20 | UINT | Optional |
| Decimal Precision | 21 | USINT | Optional |
| International Parameter Name | 22 | STRINGI | Optional |
| International Engineering Units | 23 | STRINGI | Optional |
| International Help String | 24 | STRINGI | Optional |

1 See Maximum/Minimum Attribute Semantics in the Parameter Object definition in Table 5A-14.11 for further information.

2 The data type for this field deviates from the Parameter Object, which is specified as an INT. If the ParamN entry is referencing a parameter that exists in a Parameter Object within a device, this value will need to be kept in the INT range.

3 See Table C-6.1 for valid values.

4 The Data Type of Field Numbers 10 and 11 are based on the value of Field Number 5. (In the bullets below, the first item is the Field Number 5 value and the second item is the Field Number 10 and 11 date type).

- STRING - UINT
- STRING2 - UINT
- STRINGN - UINT
- SHORT_STRING - USINT
- STRINGI - UINT
- EPATH - UINT
- ENGUNIT – empty
- All others – data type specified in Data Type field (field 5)

5 Data type is the value specified in Data Type field (field 5)

The following ParamN fields, which require additional definition (beyond the corresponding Parameter Object instance attribute definition), are listed below.

#### 7-3.6.6.1.1 Reserved, Field Number 1

This first field shall contain a zero.

#### 7-3.6.6.1.2 Link Path Size, Field Number 2

Leave this field empty or set to 0 if the Link Path is either empty or contains a null string.

#### 7-3.6.6.1.3 Link Path, Field Number 3

The link path identifier. If the Not Addressable descriptor bit value is 1 or the Link Path is empty, the attribute described by this ParamN is not directly addressable from the network. If the Not Addressable descriptor bit value is 0 and this field is a null string, “”, it shall be addressable as the data attribute (instance attribute 1) of the Nth instance of the Parameter object.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.6.1.4 Data Size, Field Number 6

The data size value for variable size data types shall be as follows:

- STRING – 1
- STRING2 – 2
- SHORT_STRING – 1
- EPATH – empty
- STRINGI – empty

#### 7-3.6.6.1.5 Parameter Name, Field Number 7

The textual parameter name. This field may be longer than the limit specified in the Parameter Object. A tool may truncate this text to the limit specified in the Parameter Object.

#### 7-3.6.6.1.6 Units String, Field Number 8

The textual display units character array. This field may be longer than the limit specified in the Parameter Object. A tool may truncate this text to the limit specified in the Parameter Object.

#### 7-3.6.6.1.7 Help String, Field Number 9

The textual help character array. This field may be longer than the limit specified in the Parameter Object. A tool may truncate this text to the limit specified in the Parameter Object.

#### 7-3.6.6.1.8 Multiplier, Divisor, Base and Offset Links, Field Number 17 - 20

ParamN identifier. This field value specifies N of the ParamN whose value is used for link scaling.

### 7-3.6.6.2 Enum Keyword

The "**Enum**" keyword shall be used to provide an enumeration list of parameter choices to present to the user. The entry keyword for all enumerated parameters shall consist of the character array, "**Enum**", combined with the decimal number from the corresponding Param entry. Any Param entry whose type is integer (BOOL, SINT, INT, DINT, LINT, USINT, UINT, UDINT or ULINT) or bit string (BYTE, WORD, DWORD or LWORD) can be enumerated using the "**Enum**" keyword. If the corresponding Param entry Descriptor field, bit 8 (Allows both enumeration and min/max range values) is set, then values in the min/max range are also valid. If the corresponding Param entry Descriptor bit 8 is set and an Enum entry is within the min/max range, the Enum string shall take precedence.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 7-3.26 EnumN Entry Keyword Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| First Enum | 1 | For integer data types, the data type of associated ParamN.<br>For bit string data types, USINT | Required |
| First Enum String | 2 | STRING | Required |
| Nth Enum | 3,5,7,… | For integer data types, the data type of associated ParamN.<br>For bit string data types, USINT | Optional |
| Nth Enum String | 4,6,8,… | STRING | Conditional[1] |

Table Footnotes
1 Required if preceding field is specified, not allowed if preceding field is not specified.

- First Enum – for integer data types this is the value of the first enumerated item of the ParamN entry. For bit string data types this is the bit number (zero based) of the first enumerated item of the ParamN entry.
- First Enum String – string associated with first Enum.
- Nth Enum – for integer data types this is the value of the Nth enumerated item of the ParamN entry. For bit string data types this is the bit number (zero based) for the Nth enumerated item of the ParamN entry.
- Nth Enum String – string associated with Nth Enum.

### 7-3.6.6.3 Fixed Keyword

The "**Fixed**" keyword shall be used to provide information for Param entries whose data type is bit string (BYTE, WORD, DWORD, LWORD), for bit values that are not selectable. The entry keyword shall consist of the character array, "**Fixed**", combined with the decimal number from the corresponding Param entry.

**Table 7-3.27 FixedN Entry Keyword Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| First fixed bit number | 1 | USINT | Required |
| First fixed bit number value | 2 | BOOL | Required |
| Nth fixed bit number | 3, 5, 7, … | USINT | Optional |
| Nth fixed bit number value | 4, 6, 8, … | BOOL | Conditional [1] |

1 Required if preceding field is specified, not allowed if preceding field is not specified

- First fixed bit number – the bit number (zero based) of the first fixed bit number of the ParamN entry.
- First fixed bit number value – the value the first fixed bit number must be set to (0 or 1).
- Nth fixed bit number – the bit number (zero based) of the Nth fixed bit number of the ParamN entry.
- Nth fixed bit number value – the value the Nth fixed bit number must be set to (0 or 1).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.6.6.4 ConstructedParam Keyword

The "ConstructedParam" keyword shall be used to represent a Constructed Data Type (0xB0), see C-6.2.3, Defined Constructed Data Types. The entry keyword shall be one of the following character arrays, "**ConstructedParam**", "**ProxyConstructedParam**", "**ProxiedConstructedParam**", combined with the Parameter object instance number (decimal) for the device, e.g. "**ConstructedParam1**". When the "**ConstructedParamN**" does not exist within a node then the allowed number range of N shall be identical to the UDINT data type, except that 0 is not allowed. The actual parameter object instance may, but need not be, implemented in the device. Conversely, it is not required that ALL parameter object instances have a corresponding "**ConstructedParamN**" entry in an EDS. Commas shall separate all fields, and a semicolon shall indicate the end of an entry. Each entry contains the formatted fields shown in the table below. The "**ProxyConstructedParam**"and "**ProxiedConstructedParam**" keywords are defined further in Chapter 7-3.7, Modular EDS File Requirements.

**Table 7-3.28 ConstructedParamN Entry Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Link Path | 1 | Padded EPATH | Conditional |
| Descriptor | 2 | WORD | Conditional |
| Data Type Code | 3 | WORD | Required |
| Data Size | 4 | UINT | Required |
| Name | 5 | STRING | Required |
| Units String | 6 | STRING | Conditional [2] |
| Member Unit String | 7 | {STRING, STRING, …} [3] | Conditional [2] |
| Help String | 8 | STRING | Required |
| Member Help String | 9 | {STRING, STRING, …} [3] | Optional |
| Minimum Value | 10 | See Data Type Code field [1] | Optional |
| Maximum Value | 11 | See Data Type Code field [1] | Optional |
| Default Value | 12 | See Data Type Code field [1] | Required |
| Type Specific Field Count | 13 | UINT | Conditional [4] |
| Type Specific Field | 14, 15, … | See Constructed Data Type Definition [5] | Conditional [6] |

Table Footnotes

1. The Data Type of Field Numbers 10, 11 and 12 are based on the value of Field Number 3.
2. At least one of these fields must be specified.
3. The number of STRINGs is determined by the number of members in the Constructed Data Type.
4. The Type Specific Field Count shall be present if the Constructed Data Type defines Type Specific fields otherwise this field is not allowed.
5. The Data Type of Fields 14, 15 and so on is based on the definition of the Constructed Data Type indicated by the Data Type Code in Field 3. See section 7-3.5.2 Defined Constructed Data Type Encodings for type specific definition.
6. Field is required if Field 13 is not equal to zero. The number of Type Specific Fields is dependent on the value of field 13.

#### 7-3.6.6.4.1 Link Path, Field Number 1

This is the link path identifier. If the Link Path is empty, the attribute described by this ConstructedParamN is not directly addressable from the network.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.6.4.2 Descriptor, Field Number 2

See Parameter Object definition of Descriptor attribute, 5A-14.2.1.1. Only bits 4,5,9,11 and 14 apply all others must be 0. If the Descriptor is empty a value of 0 is used.

#### 7-3.6.6.4.3 Data Type Code, Field Number 3

This field represents the Data Type Code described in Appendix C-6.2.3 Defined Constructed Data Types.

#### 7-3.6.6.4.4 Data Size, Field Number 4

This is the size of this Constructed Data Type in bytes. If the value is 0 the Constructed Data Type size is variable. The data size may be used by tools that do not have inherent knowledge of the Constructed Date Type to provide raw access.

#### 7-3.6.6.4.5 Name, Field Number 5

The textual name of the item (e.g. attribute name) represented by this keyword.

#### 7-3.6.6.4.6 Units String, Field Number 6

The Units String represents the Defined Constructed Data Type as a whole. (i.e. “Widgets per Hour”)

#### 7-3.6.6.4.7 Member Units String, Field Number 7

The Member Units String represents each member of the Defined Constructed Data Type. To represent the members use the following format {“unit for member 1”, “unit for member 2”,…””, “unit for member n”}.

#### 7-3.6.6.4.8 Help String, Field Number 8

The Help String represents the Defined Constructed Data Type as a whole. (i.e. “Widgets Help”)

#### 7-3.6.6.4.9 Member Help String, Field Number 9

The Member Help String represents each member of the Defined Constructed Data Type. To represent the members use the following format {“help for member 1”, “help for member 2”,… “”,“help for member n”}.

#### 7-3.6.6.4.10 Minimum Value, Field Number 10

The minimum value this Constructed Data Type can be set to. If empty the lowest possible value for the Constructed Data Type is supported.

#### 7-3.6.6.4.11 Maximum Value, Field Number 11

The maximum value this Constructed Data Type can be set to. If empty the highest possible value for the Constructed Data Type is supported.

#### 7-3.6.6.4.12 Default Value, Field Number 12

The actual value this Constructed Data Type is set to initially.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.6.4.13 Type Specific Field Count, Field Number 13

The number of type specific entries to follow.

#### 7-3.6.6.4.14 Type Specific Field(s) , Field Numbers 14, 15…

The use of this field or fields is determined by the Data Type Code, see section 7-3.5.2 Defined Constructed Data Type Encodings for type specific details.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-3.11 Sample Electronic Data Sheet Illustrating the Parameters Section**

```
[File]
   ...
[Device]
   ...
[ParamClass]
   ...
[Params]
   Param1 = 0,
    6,"20 04 24 68 30 03",  $ Link Path Size, Link Path
    0x0002,                 $ Descriptor
    0xD1, 1,                $ BYTE Data Type, Data Size
    "Idle Action",          $ Name
    "",                     $ Units
    "",                     $ Help string
    ,,0,                    $ Min/Max/Default
    ,,,,                    $ Scaling
    ,,,,
    ;
   Enum1 =
    0, "Fault",
    1, "Stop",
    2, "Zero Data",
    3, "Hold Last",
    4, "Send Flt Cfg”;
   Fixed1 =
    1, 0,         $ The idle action is never stop
    4, 1;         $ The fault configuration is always sent

   Param2 =       $ not addressable from link
   0, , , 0x0082, 0xC6, 1, "speed control", "", "", 3, 12, 3, ,,,,,,,,;
   Enum2  = 3, "stop", 8, "slow", 12, "fast";

   ConstructedParam1=
   “20 4e 24 01 30 09”,            $ Link Path
   0x0000,                         $ Descriptor
   1,10,                           $ SIGNED_ODOMETER, Data Size
   "Total Energy",                 $ Name
   "kWh",                          $ Units
   {“TWh”,”GWh”,”MWh”,”kWh”,”Wh”}, $ Member Units
   "Total energy in kWh",          $ Help string
   {“This is Terawatt-Hours”,
    “This is Gigawatt-Hours”,
    “This is Megawatt-Hours”,
    “This is Kilowatt-Hours”,
    “This is Watt-Hours”},         $ Member Help
   {-999,-999,-999,-999,-999},     $ Min
   {999,999,999,999,999,999},      $ Max
   {0,0,0,0,0}                     $ Default
   1,                              $ 1 Type Specific Field follows
   -3;                            $ Decimal point location between Wh and kWh

[Groups]
   ...
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.6.7 Parameter Groups Section

The parameter groups section begins with the keyword [Groups] and identifies all of the parameter groups in a device.

Table 7-3.29 shows the Common Object Entry Keywords overridden by the Groups Section.

**Table 7-3.29 Common Object Entry Keyword Overrides in the Groups Section**

| Entry Name | Entry Keyword | Required/Optional |
|---|---|---|
| Revision | Revision | Optional [1] |
| Maximum Instance Number | MaxInst | Optional [2] |
| Number of Static Instances | Number_Of_Static_Instances | Optional [2] |
| Maximum Number of Dynamic Instances | Max_Number_Of_Dynamic_Instances | Optional [2] |

Table Footnotes

1. If this optional entry is missing, the revision of the Parameter Group object shall be 1.
2. This keyword, although defined in section 7-3.6.1.3, Common Object Class Entry Keywords as Required, is Optional in the [Groups] section.

Table 7-3.30 shows the Addition Entry Keywords overridden in the Groups Section.

**Table 7-3.30 Entry Keywords in the Groups Section**

| Entry Name | Entry Keyword | Required/Optional |
|---|---|---|
| Group | GroupN | Optional |

#### 7-3.6.7.1 Group Type Entry Keywords

Each parameter group contains a list of the parameter object instances in the group. The entry keyword for each group consists of the combination of the character array, "**Group**", and the Parameter Group instance number (decimal) in the device, for example "**Group1**". The decimal numbers must start at one and increment by one.

The fields in each entry contain the group name, the number of members in the group, and the instance numbers of the parameter objects in the group or **VariantN/VariantExaN** where N is the variant number in this EDS or ProxyN where N is the ProxyParam number in this EDS. A comma separates all fields, and a semicolon indicates the end of the entry. The parameter group section contains:

**Table 7-3.31 GroupN Entry Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Group Name String | 1 | STRING | Required |
| Number of Members | 2 | UINT | Required |
| Parameter, Proxy Parameter or Variant | 3 thru (number of members + 2) | UINT for Param, ProxyN for Proxy Param or VariantN/VariantExaN for Variant | Required |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-3.12 Sample Electronic Data Sheet Illustrating the Parameter Groups Section**

```
$
[File]
   ...
[Device]
   ...
[ParamClass]
   ...
[Params]
   ...
[Groups]
   Group1 = "Setup", 3, 1, Variant2, Proxy1; $ group 1 named Setup consists
                                             $ of Param1, Variant2 & ProxyParam1

   Group2 = "Monitor", 3, 2, 3, Proxy4; $ group 2 named Monitor consists of
                                        $ Param2, Param3 and ProxyParam4

   Group3 = "Maintenance", 2, 1, 3;     $ group 3 named Maintenance
                                        $ consists of Param1 and Param3
```

### 7-3.6.8 Assembly Section

The Assembly section begins with the keyword [Assembly] and describes the structure of a data block. Often this block is the data attribute of an Assembly object; however, this section of the EDS can be used to describe any complex structure. The description of this data block parallels the mechanism that the Assembly object uses to describe its member list.

Table 7-3.32 shows the Common Object Entry Keywords overridden by the Assembly Section.

**Table 7-3.32 Common Object Entry Keyword Overrides in the Assembly Section**

| Entry Name | Entry Keyword | Required/Optional |
|---|---|---|
| Revision | Revision | Optional [1] |
| Maximum Instance Number | MaxInst | Optional [2] |
| Number of Static Instances | Number_Of_Static_Instances | Optional [2] |
| Maximum Number of Dynamic Instances | Max_Number_Of_Dynamic_Instances | Optional [2] |

Table Footnotes

1. If this optional entry is missing, the revision of the Assembly object shall be 2.
2. This keyword, although defined in section 7-3.6.1.3, Common Object Class Entry Keywords as Required, is Optional in the [Assembly] section.

Table 7-3.33 shows the additional Entry Keywords in the Assembly Section.

**Table 7-3.33 Entry Keywords in the Assembly Section**

| Entry Name | Entry Keyword | Required/Optional |
|---|---|---|
| Assem | AssemN, AssemExaN, ProxyAssemN, ProxyAssemExaN, ProxiedAssemN, ProxiedAssemExaN | Optional |
| Variant | VariantN, VariantExaN, BitStringVariantN | Optional |
| Array | ArrayN | Optional |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.8.1 Assem type Entry Keywords

The Assem type entry keyword shall consist of a root from Table 7-3.33 above combined with the Assembly object instance number (decimal) for the device, e.g. "**Assem1**". The same instance number (decimal) shall not be used for an "**AssemN**" and an "**AssemExaN**" entry. The same instance number (decimal) shall not be used for a "**ProxyAssemN**" and a "**ProxyAssemExaN**" entry. The same instance number (decimal) shall not be used for a "**ProxiedAssemN**" and a "**ProxiedAssemExaN**" entry. If a particular instance of the Assembly object is addressable from the link, there shall be a one-for-one pairing between the **Assem** number in the EDS file and the Assembly instance number in the device if the Link Path specified is the null string ("). When the "AssemN" does not exist within a node then the allowed number range of N shall be identical to the UDINT data type, except that 0 is not allowed. Commas shall separate all fields, and a semicolon shall indicate the end of the entry. A white space or nothing between commas shall be used for optional fields not provided. The "**ProxyAssemN**", "**ProxyAssemExaN**", "**ProxiedAssemN**", and "**ProxiedAssemExaN**" keywords are defined further in Chapter 7-3.7, Modular EDS File Requirements.

"**AssemExaN**" is an extension of the "**AssemN**" keyword. "**ProxyAssemExaN**" is an extension of the "**ProxyAssemN**" keyword. "**ProxiedAssemExaN**" is an extension of the "**ProxiedAssemN**" keyword. Each "**AssemN**", "**ProxyAssemN**" and "**ProxiedAssemN**" entry shall contain the formatted fields shown in Table 7-3.34. Each "**AssemExaN**", "**ProxyAssemExaN**" and "**ProxiedAssemExaN**" entry shall contain the formatted fields shown in Table 7-3.35.

**Table 7-3.34 AssemN Entry Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Name | 1 | STRING | Optional |
| Path | 2 | Padded EPATH | Optional |
| Size | 3 | UINT | Optional |
| Descriptor | 4 | WORD | Optional |
| Reserved | 5, 6 | Empty | Not Used |
| Member Size | 7, 9, 11 … | UINT | Conditional |
| Member Reference | 8, 10, 12 … | Empty, AssemN, ParamN, UDINT or Packed EPATH | Conditional |

**Table 7-3.35 AssemExaN Entry Format**

| Field name | Field number | Data type | Required/ Optional |
|---|---|---|---|
| Name | 1 | STRING | Optional |
| Path | 2 | Padded EPATH | Optional |
| Size | 3 | UINT or ParamN | Optional |
| Descriptor | 4 | WORD | Optional |
| Reserved | 5, 6 | Empty | Not Used |
| Member Size | 7, 9, 11 … | UINT | Conditional |
| Member Reference | 8, 10, 12 … | Empty, AssemN, AssemExaN, ParamN, UDINT, VariantN, BitStringVariantN, VariantExaN, ArrayN or Packed EPATH | Conditional |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.8.1.1 Name, Field 1

The first field, called "Name", shall be a string giving a name to the data block. This optional field may be used by a user interface.

#### 7-3.6.8.1.2 Path, Field 2

The second field, called "Path", identifies the address of the data block within the device. If the Not Addressable descriptor bit value is 1 or this field is empty, the block described by this **Assem**N entry is not directly addressable from the link. If the Not Addressable descriptor bit value is 0 and this field is a null string, “”, the data block shall be addressable as the data attribute (instance attribute 3) of the Nth instance of the Assembly object.

#### 7-3.6.8.1.3 Size, Field 3

The third field, called "Size", shall be the size of the data block in bytes. When the “Size” field is specified as a ParamN, the type of the ParamN shall be UINT or USINT. When the “Member Size”/”Member Reference” fields are present and the “Size” field is present, if the “Size” field is smaller than the defined size of the corresponding “Member Size”/”Member Reference” fields, the least significant bits of the corresponding “Member Reference” fields shall be used. When the “Member Size”/”Member Reference” fields are present and the “Size” field is present, if the “Size” field is larger than the defined size of the corresponding “Member Size”/”Member Reference” fields, the entire “Member Reference” fields shall be followed by zero pads to extend the assembly to the “Size”.

When the “Member Size”/”Member Reference” fields are present and the “Size” field is not present, the defined size of the corresponding “Member Size”/”Member Reference” fields shall be used.

When the “Member Size”/”Member Reference” fields are not present and the “Size” field is not present, the size shall be 0.

#### 7-3.6.8.1.4 Descriptor, Field 4

The fourth field, “Descriptor”, shall be a bit-field that describes certain properties of the Assembly. The bits of this field are described in Table 7-3.36. If this field is empty it shall be interpreted as 0.

The Descriptor only applies to the AssemN it is defined in. In other words, the Descriptor defined in a nested AssemN overrides any parent AssemN Descriptor and the corresponding Descriptor bits defined in a nested ArrayN overrides any parent AssemN Descriptor.

**Table 7-3.36 Descriptor Field Bit Value**

| Bit | Name | Value | Meaning |
|---|---|---|---|
| 0 | Allow Value Edit | 0 | The contents of value member references shall not be edited. The member references considered to be values are a Number and an empty field. |
| | | 1 | The contents of value member references may be edited. The member references considered to be values are a Number and an empty field.<br>NOTE: This descriptor value is useful when the AssemN is used to describe an input or output assembly whose type details are dynamically defined (e.g. gateway devices). In this case the AssemN entry could contain a single Member Size/Member Reference pair, with the Member Reference being empty. A configuration tool could then allow the user to specify the detailed type information when the connection is configured or define the data as a block of octets. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Bit | Name | Value | Meaning |
|---|---|---|---|
| 1-3 | Assem Type | 0 | Not modular Assem |
| | | 1 | This Assem defines a fixed slot size, expanded adapter rack Assem, the adapter device is not included in the Assem.<br>This Assem applies to each slot in the rack, in slot order. This Assem shall specify a UINT size field, this UINT size shall be the amount of data supplied, in bytes, for each slot in the rack. Each module participating in the adapter rack Assem shall supply the same amount of data. For empty slots and slots containing modules not participating in this adapter rack Assem the pad data shall be included, the size of the pad data shall be specified by the UINT size field. The member list size may be less than or equal to the size field value. If the member list size is less than the size field value the Assem shall be padded to the size field value. |
| | | 2 | This Assem defines a fixed slot size, compressed adapter rack Assem, the adapter is not included in the Assem.<br>This Assem applies to each slot in the rack, in slot order, that is participating in this adapter rack Assem. This Assem shall specify a UINT size field, this UINT size shall be the amount of data supplied, in bytes, for each slot in the rack participating in the adapter rack Assem. The member list size may be less than or equal to the size field value. If the member list size is less than the size field value the Assem shall be padded to the size field value. |
| | | 3 | This Assem defines a variable sized, compressed adapter rack Assem, the adapter is not included in the Assem.<br>This Assem applies to each slot in the rack, in slot order, that is participating in this adapter rack Assem. This Assem may specify a ProxyParamN size field which allows each module EDS to define the amount of data included in this adapter rack Assem. |
| | | 4 | Reserved. |
| | | 5 | This Assem defines a fixed slot size, expanded adapter rack Assem, the adapter device is included in the Assem.<br>This Assem applies to each slot in the rack, in slot order. This Assem shall specify a UINT size field, this UINT size shall be the amount of data supplied, in bytes, for each slot in the rack. Each module participating in the adapter rack Assem shall supply the same amount of data. For empty slots and slots containing modules not participating in this adapter rack Assem the pad data shall be included, the size of the pad data shall be specified by the UINT size field. The member list size may be less than or equal to the size field value. If the member list size is less than the size field value the Assem shall be padded to the size field value. |
| | | 6 | This Assem defines a fixed slot size, compressed adapter rack Assem, the adapter is included in the Assem.<br>This Assem applies to each slot in the rack, in slot order, that is participating in this adapter rack Assem. This Assem shall specify a UINT size field, this UINT size shall be the amount of data supplied, in bytes, for each slot in the rack participating in the adapter rack Assem. The member list size may be less than or equal to the size field value. If the member list size is less than the size field value the Assem shall be padded to the size field value. |
| | | 7 | This Assem defines a variable sized, compressed adapter rack Assem, the adapter is included in the Assem.<br>This Assem applies to each slot in the rack, in slot order, that is participating in this adapter rack Assem. This Assem may specify a ProxyParamN size field which allows each module EDS to define the amount of data included in this adapter rack Assem. |
| 4 | Not Addressable | 0 | If the Path field is not empty, the block described by this assembly entry is directly addressable from the link via the Set_Attribute and Get_Attribute services. |
| | | 1 | The block described by this assembly entry is not directly addressable from the link via the Set_Attibute or Get_Attribute services. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Bit | Name | Value | Meaning |
|---|---|---|---|
| 5-7 | NULL Indirect Parameter Handling | 0 | If a member reference is a Parameter whose Indirect Parameter Reference bit is set and whose referenced parameter value is 0 (NULL), then the member size and member reference pair is ignored. |
| | | 1 | If a member reference is a Parameter whose Indirect Parameter Reference bit is set and whose referenced parameter value is 0 (NULL), then the member size and member reference pair and any following member size and member reference pairs are ignored. |
| | | 2 | If a member reference is a Parameter whose Indirect Parameter Reference bit is set and whose reference parameter value is 0 (NULL), then the number of bits specified by the member size field, if specified, shall be used as a pad within the Assembly object, if the member size field is not specified the pad size shall be 16 bits. [3] |
| | | 3-7 | Reserved |
| 8-10 | Write Allowed [1] | 0 | Indicates the assembly may or may not be able to be written when any I/O connections are open. |
| | | 1 | Indicates the assembly can be written when any I/O connections are open. If this assembly contains member references that are assemblies, the referenced assemblies must have their write allowed value set to 0 or 1. |
| | | 2 | Indicates the assembly can not be written when the I/O connection associated with this assembly is open. If this assembly contains member references that are assemblies, the referenced assemblies must have their write allowed value set to 0, 1 or 2. |
| | | 3 | Indicates the assembly can not be written when any I/O connections are open. |
| | | 4-7 | Reserved. |
| 11 | Indirect Parameter Handling [2] | 0 | If a member reference is a Parameter whose Indirect Parameter Reference bit is set, the Parameter shall be handled as an indirect Parameter (i.e treat as a pointer). The size of the Parameter being pointed to shall be used as the Member Size, if the Member Size not specified. |
| | | 1 | If a member reference is a Parameter whose Indirect Parameter Reference bit is set, the Parameter shall be handled as a direct Parameter (ignore the Indirect Parameter Reference bit). The size of the Parameter (the pointer) shall be used as the Member Size, if Member is not specified. |
| 12-15 | Reserved | 0 | |

Table Footnotes:

1 A write to the assembly can fail due to conditions other than I/O connections being open, like object state conflict.

2 If an Assem contains multiple Parameters whose Indirect Parameter Reference bit is set and some Parameters should be treated as direct Parameters and others as indirect Parameters, sub Assems can be created with different values for the Indirect Parameter Handling descriptor bit.

3 Devices that want to support variable size pads should provide pad Parameters for the user to select as the value of Indirect Parameters.

See section 7-3.6.8.5, Examples of Adapter/Rack Assem Data Formatting for examples.

See section 7-3.6.8.6 Example of Indirect Parameter Handling for examples.

#### 7-3.6.8.1.5 Reserved Field 5

Field 5 shall be reserved for future definition and shall be empty.

#### 7-3.6.8.1.6 Reserved Field 6

Field 6 shall be reserved for future definition and shall be empty.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.8.1.7 Member Size/Member Reference Fields

The remaining fields shall be paired such that a "Member Size" field is paired with a "Member Reference" field making the total number of fields even. Zero remaining fields is valid, which represents an empty assembly. The pairs shall correspond to an Assembly object member list.

The allowed entries for the "Member Reference" field shall be one of the following:

- a ParamN entry from the [Params] section;
- an AssemN entry from the [Assembly] section;
- an AssemExaN entry from the [Assembly] section;
- a VariantN entry from the [Assembly] section;
- a VariantExaN entry from the [Assembly] section;
- a BitStringVariantN entry from the [Assembly] section;
- a packed EPATH;
- a UDINT;
- an empty field;

If the "Member Reference" field is empty, the number of bits specified by the "Member Size" field shall be used as a pad within the Assembly object. A "Member Reference" field containing a null string shall be treated as if the field was empty. A "Member Reference" field and its corresponding "Member Size" shall not both be empty. The member reference specifying an EPATH shall consist of either Logical or Data Segments.

The "Member Size" field shall have units of bits. If a "Member Size" field is empty, the defined size of the corresponding "Member Reference" field shall be used. The defined size of a **Param** entry shall be as given in its 6$^{th}$ field (size). The defined size of an **Assem** entry shall be as given in its 3$^{rd}$ field (size).

The members shall be placed into the data block least significant bit first just as they are in the Assembly object. If a "Member Size" field is smaller than the defined size of the corresponding "Member Reference" field, the least significant bits of the corresponding "Member Reference" field shall be used. If a "Member Size" field is larger than the defined size of the corresponding "Member Reference" field, the entire member shall be followed by zero pads to extend the member to the "Member Size". The data block represented shall be an integer number of bytes. For example, **Assem5** in Figure 7-3.13 shall be 1 byte long and have a default value of 0x21.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-3.13 [Assembly] Section Example**

```
[Params]
   Param1 =
   0,                        $ first field shall equal 0
   6, "20 0F 24 01 30 01",   $ path size, path
   0x0000,                   $ descriptor
   0xD2,                     $ data type : 16-bit WORD
   2,                        $ data size in bytes
   "Idle state",             $ name
   "",                        $ units
   "User Manual p48",        $ help string
   0, 2, 1,                  $ min, max, default data values
   0, 0, 0, 0,               $ mult, dev, base, offset scaling not used
   0, 0, 0, 0,               $ mult, dev, base, offset link not used
   0;                        $ decimal places not used

   Param2 = 0, 6, "20 0F 24 02 30 01",     $ path size, path
            0x0000, 0xD2, 2, "Fault state", "", "User Manual p49",
            0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0;
[Assembly]
   Revision = 2;

   Assem5 = "configuration", "20 04 24 05 30 03",1,,,,
            4, Param1,
            3, Param2,
            1, ;
```

### 7-3.6.8.1.8 Specifying Bit Fields

The ParamN:b syntax is used to specify which bits from a parameter are to be included in an assembly. The ':b' indicates the bit offset into the parameter, where '0' refers to bit 0. The member reference size indicates the number of relevant bits. It is invalid to specify a size and bit offset that 'overflow' the parameter. For example, the assembly member reference "16, Param1:2" is not valid when the parameter is 16 bits in size. The bits used from the parameter start at the offset specified and continue with the more significant bits for the size specified. This syntax allows disjoint bits within an assembly to be represented with a single parameter.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-3.14 ParamN:b Syntax Example**

```
[Params]
   Param1 = 0,                $ first field shall equal 0
   ,,                         $ path size, path
   0x0002,                    $ descriptor
   0xC6,                      $ data type: Unsigned short integer
   1,                         $ data size in bytes
   "Channel 0 Range Selection Bits",   $ name
   "",                        $ units
   "",                        $ help string
   0,3,0,                     $ min, max, default data values
   0,0,0,0,                   $ mult, dev, base, offset scaling not used
   0,0,0,0,                   $ mult, dev, base, offset link not used
   0;                         $ decimal places
   Enum1 = 0, “Off”, 1, “4-20mA”, 2, “0-10V”, 3, “-10 to +10V”

$ Param2, Param3 & Param4 are similar for Channel 2, Channel 3 & Channel 4

[Assembly]
  Assem1 = “Configuration Data”, "20 7D 24 00 30 0C",,,,,
  1,Param1:0,               $ Channel 0 full range bit
  1,Param2:0,               $ Channel 1 full range bit
  1,Param3:0,               $ Channel 2 full range bit
  1,Param4:0,               $ Channel 3 full range bit
  4,,                       $ Not used – must be 0 [fill bits]
  1,Param1:1,               $ Channel 0 selection bit
  1,Param2:1,               $ Channel 1 selection bit
  1,Param3:1,               $ Channel 2 selection bit
  1,Param4:1,               $ Channel 3 selection bit
  4,;                       $ Not used – must be 0
```

### 7-3.6.8.2 Variant, VariantExa and BitStringVariant Entry Keywords

The “**Variant**”, “**VariantExa**” and "**BitStringVariant**" keywords shall be used to describe a dependent member of an assembled data buffer. The “**Variant**” and “**VariantExa**” keywords are similar to a ‘Switch’ statement in a high-level programming language. The ‘Switch’ statement is a multi-way decision that tests whether an expression matches one of a number of constant integer values, and branches accordingly. The “**Variant**” or “**VariantExa**” entry in the [Assembly] section serves a similar purpose. It uses specific bits in an assembly or a parameter to obtain a value called the ‘Switch selector’ that is used to match a number of constant integer values. Once a match is found the corresponding assembly or parameter is used in place of the “**VariantN**” or “**VariantExa**” entry (see Figure 7-3.15 for an example). If multiple matches are found one of the corresponding assemblies or parameters shall be used in place of the “**VariantN**” or “**VariantExa**” entry. The method for selecting which corresponding assembly or parameter requires user intervention. Multiple matches are supported because determination of the device configuration is not available by electronic means.

The “**BitStringVariant**” keyword is used to select 0, 1 or multiple members based on the bits set or reset in the ‘Bit switch selector’. It uses specific bits in an assembly or a parameter to obtain a bit string value called the ‘Bit switch selector’ that is used to select a number of constant bit number values. All matches found select the corresponding assembly or parameter and they are used in place of the “**BitStringVariantN**” entry.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The entry keyword shall consist of the character array, "**Variant**", "**VariantExa**" or "**BitStringVariant**", combined with a decimal number, e.g. "**Variant1**", "**VariantExa1**" or "**BitStringVariant1**". The same decimal number shall not be used for a "**VariantN**" and a "**VariantExaN**" entry. The allowed number range of N shall be identical to the UDINT data type, except that 0 is not allowed. Commas shall separate all fields, and a semicolon shall indicate the end of the entry. White space or nothing between commas shall be used for optional fields not provided. Each "**BitStringVariantN**" entry shall contain the formatted fields shown in Table 7-3.37 .

**Table 7-3.37 BitStringVariantN Entry Format**

| Field name | Field number | Data type | Required/Optional |
|---|---|---|---|
| Name | 1 | STRING | Optional |
| Help String | 2 | STRING | Optional |
| Reserved | 3 | Reserved | Not used |
| Reserved | 4 | Reserved | Not used |
| Reserved | 5 | Reserved | Not used |
| Bit switch selector | 6 | ParamN, AssemN or AssemExaN | Required |
| First bit selection value | 7 | UINT | Required |
| First bit set selection entry | 8 | Empty, ParamN, AssemN, AssemExaN or ArrayN | Required |
| First bit reset selection entry | 9 | Empty, ParamN, AssemN, AssemExaN or ArrayN | Required |
| | | | |
| | | | |
| Subsequent bit selection value | 10, 13, … | UINT | Optional |
| Subsequent bit set selection entry | 11, 14, … | Empty, ParamN, AssemN, AssemExaN or ArrayN | Optional |
| Subsequent bit reset selection entry | 12, 15, … | Empty, ParamN, AssemN, AssemExaN or ArrayN | Optional |

"**VariantExaN**" is an extension of the "**VariantN**" keyword. Each "**VariantN**" entry shall contain the formatted fields shown in Table 7-3.38. Each "**VariantExaN**" entry shall contain the formatted fields shown in Table 7-3.39.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 7-3.38 VariantN Entry Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Name | 1 | STRING | Optional |
| Help String | 2 | STRING | Optional |
| Reserved | 3 | Empty | Not Used |
| Reserved | 4 | Empty | Not Used |
| Reserved | 5 | Empty | Not Used |
| Switch selector | 6 | ParamN or AssemN | Required |
| First Selection value | 7 | UINT | Required |
| First Selection entry | 8 | ParamN | Required |
| Second Selection value | 9 | UINT | Required |
| Second Selection entry | 10 | ParamN | Required |
| Subsequent Selection values | 11,13,… | UINT | Optional |
| Subsequent Selection entries | 12,14,… | ParamN | Optional |

**Table 7-3.39 VariantExaN Entry Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Name | 1 | STRING | Optional |
| Help String | 2 | STRING | Optional |
| Reserved | 3 | Reserved | Not Used |
| Reserved | 4 | Reserved | Not Used |
| Reserved | 5 | Reserved | Not Used |
| Switch selector | 6 | ParamN, AssemN or AssemExaN | Required |
| First Selection value | 7 | UINT | Required |
| First Selection entry | 8 | ParamN, AssemN, AssemExaN or ArrayN | Required |
| Second Selection value | 9 | UINT | Required |
| Second Selection entry | 10 | ParamN, AssemN, AssemExaN or ArrayN | Required |
| Subsequent Selection values | 11,13,… | UINT | Optional |
| Subsequent Selection entries | 12,14,… | ParamN, AssemN, AssemExaN or ArrayN | Optional |

#### 7-3.6.8.2.1 Name, Field 1

The “Name” field shall be a string giving a name to the variant. This optional field may be used by the user interface.

#### 7-3.6.8.2.2 Help String, Field 2

The “Help String” field shall be a string providing help for the variant. This optional field may be used by the user interface.

#### 7-3.6.8.2.3 Reserved Field 3

Field 3 shall be reserved for future definition and shall be empty.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.8.2.4 Reserved Field 4

Field 4 shall be reserved for future definition and shall be empty.

#### 7-3.6.8.2.5 Reserved Field 5

Field 5 shall be reserved for future definition and shall be empty.

#### 7-3.6.8.2.6 Switch Selector or Bit Switch Selector, Field 6

The syntax “xxx:N1:N2” may be used where xxx is a ParamN, AssemN or AssemExaN and N1 specifies a bit offset into “xxx” and N2 specifies the number of bits to use. For example, “Assem1:0:4” indicates to use the first four bits of Assem1. The assembly or parameter specified as the “Switch selector” or "Bit switch selector" shall exist within the device. The switch selector field shall not specify a portion of an assembly that is a variant.

If the “Switch selector” or “Bit switch selector” exists in the same assembly as the variant that specifies the “Switch selector” or “Bit switch selector”, then the “Switch selector” or “Bit switch selector” shall precede the variant in the assembly.

The “Bit switch selector” must evaluate to a bit string. If the “Bit switch selector” is a ParamN the type of the ParamN must be BYTE, WORD, DWORD or LWORD types. If the “Bit switch selector” is an AssemN or AssemExaN the members must be of type BYTE, WORD, DWORD or LWORD.

#### 7-3.6.8.2.7 Switch Selection Entry/Value Fields

The remaining fields shall be paired such that a “Selection value” field is paired with a “Selection entry” field. The “Selection value” field is used in the matching process described below. There is no numeric ordering of the pairs required. Only the matched selection entry specified in a VariantN or VariantExaN shall be visible through the variant, other selection entry(s) in the VariantN or VariantExaN shall not be visible.

The “Switch selector” field is used to determine a specific “Selection value”. The “Switch selector” value is compared with each of the “Selection value” fields until a match is found. The “Selection entry” paired with the matched “Selection value” is then used as the entry (parameter or assembly) in the assembly. If a match is not found the user interface shall display an error.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-3.15 Variant Keyword Example**

```
[Params]
   Param1 =
   0,                          $ first field shall equal 0
   ,,                          $ path size, path
   0x0002,                     $ descriptor
   0xC6                        $ data type : Unsigned short integer
   1,                          $ data size in bytes
   "Output Channel 0 Configuration",    $ name
   "",                         $ units
   "",                         $ help string
   0,2,0,                      $ min, max, default data values
   0,0,0,0,                    $ mult, dev, base, offset scaling not used
   0,0,0,0,                    $ mult, dev, base, offset link not used
   0;                          $ decimal places not used
   Enum1 = 0, “Channel not configured”, 1, “4-20mA”, 2, “+-10V”;

   Param2 =
   0,                          $ first field shall equal 0
   ,,                          $ path size, path
   0x0044,                     $ descriptor (scaling and decimal precision)
   0xC7,                       $ data type : 16 bit unsigned integer
   2,                          $ data size in bytes
   “Output Channel 0 4-20mA ", $ name
   "mA",                       $ units
   "",                         $ help string
   0,30840,0,                  $ min, max, default data values
   1,1927,100,7712,            $ mult, dev, base, offset
   0,0,0,0,                    $ mult, dev, base, offset link not used
   2;                          $ decimal places

   Param3 =
   0,                          $ first field shall equal 0
   ,,                          $ path size, path
   0x0044,                     $ descriptor (scaling and decimal precision)
   0xC7,                       $ data type : 16 bit unsigned integer
   2,                          $ data size in bytes
   "Output Channel 0 +-10V ",  $ name
   "V",                        $ units
   "",                         $ help string
   31969,33567,31969,          $ min, max, default data values
   1,79,100,-32769,            $ mult, dev, base, offset
   0,0,0,0,                    $ mult, dev, base, offset link not used
   2;                          $ decimal places

   Param4 =
   0,                          $ first field shall equal 0
   ,,                          $ path size, path
   0x0044,                     $ descriptor (scaling and decimal precision)
   0xC7,                       $ data type : 16 bit unsigned integer
   2,                          $ data size in bytes
   "Output Channel 0 0-20mA ", $ name
   "mA",                       $ units
   "",                         $ help string
   0, 31208,0,                 $ min, max, default data values
   1,1560,1000,0,              $ mult, dev, base, offset
   0,0,0,0,                    $ mult, dev, base, offset link not used
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
   2;                          $ decimal places

[Assembly]
   Assem1 =
   “configuration”,”20 7D 24 00 30 0C”, ,,,,
   4,Param1;                   $ Output Channel 0 Configuration

   Variant1 = “Output Channel 0 Configuration”,      $ name
   ,,,,                        $ Reserved fields
   Assem1:0:4,                $ value of Param1 in Assem1 is used to select
                               $ one of the following parameters
   0x01, Param2,               $ if the above value is 0x01 then use Param2
   0x02, Param3,               $ the user needs to indicate which parameter
                               $ to user (Param3 or Param4) when the above
                               $ value is 2
   0x02, Param4;               $ the user needs to indicate which parameter
                               $ to user (Param3 or Param4) when the above
                               $ value is 2
   Assem2 =
   “output” ,,,,,,
   4, Variant1;
```

### 7-3.6.8.2.8 Bit Selection Value/Bit Set Selection Entry /Bit Reset Selection Entry Fields

The remaining fields shall be in sets of three such that a “Bit selection value” field is combined with a “Bit set selection entry” field and a “Bit reset selection entry” field. The “Bit selection value” field is used in the matching process described below.

The “Bit selection value” field shall be unique, duplicate values are not allowed.

There is no numeric ordering of the “Bit selection value” entries required. Only the matched selection entries specified in a BitStringVariantN shall be included through the variant, other selection entry(s) in the BitStringVariantN shall not be included.

The “Bit switch selector” field is used to determine a specific “Bit selection value”. The bits in the “Bit switch selector” value are compared with each of the “Bit selection value” fields; the “Bit selection value” specifies a bit number. If the bit in the “Bit switch selector” is set the “Bit selection value” field that specifies that bit number is used to select the corresponding “Bit set selection entry” field. If the bit in the “Bit switch selector” is reset the “Bit selection value” field that specifies that bit number is used to select the corresponding “Bit reset selection entry” field.

All “Bit selection entry” fields combined with the matched “Bit set selection entry”/”Bit reset selection entry” fields are used as the entry (parameter or assembly) in the BitStringVariant. The fields are combined in the order they are specified.

Figure 7-3.16 shows an example of a BitStringVariant.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-3.16 BitStringVariant Keyword Example**

```
[Params]
   Param1 =
           0,                            $ first field shall equal 0
           7, "20 93 25 1A 40 30 09",    $ path size, path
           0x02,                         $ descriptor
           0xD1,                         $ data type: BYTE
           1,                            $ data size in bytes
           "M-S Output", "",             $ name, help string
           "",                           $ units
           0x01, 0x10, 0x1F,             $ min, max, default data values
           ,,,,                          $ scaling, not used
           ,,,,                          $ link scaling, not used
           ;                             $ decimal places, not used

   Enum1 =
           0, "Set - Status/Fdbk",
           1, "Set - DL A Out, Reset – DL B Out",
           3, "Reset - DL C Out",
           4, "Set - DL D Out";

   Param10 =
           0,                            $ first field shall equal 0
           ,,                            $ path size, path
           0x00,                         $ descriptor
           0xC7,                         $ data type: UINT
           2,                            $ data size in bytes
           "Logic Status", "",           $ name, help string
           "",                           $ units
           0, 65535, 0,                  $ min, max, default data values
           ,,,,                          $ scaling, not used
           ,,,,                          $ link scaling, not used
           ;                             $ decimal places, not used

   Param11 =
           0,                            $ first field shall equal 0
           ,,                            $ path size, path
           0x00,                         $ descriptor
           0xC3,                         $ data type: INT
           2,                            $ data size in bytes
           "Feedback", "",               $ name, help string
           "",                           $ units
           -32767, 32767, 0,             $ min, max, default data values
           ,,,,                          $ scaling, not used
           ,,,,                          $ link scaling, not used
           ;                             $ decimal places, not used
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
Param12 =
        0,                          $ first field shall equal 0
        ,,                          $ path size, path
        0x00,                       $ descriptor
        0xC8,                       $ data type: UDINT
        4,                          $ data size in bytes
        "Data Out A1", "",          $ name, help string
        "",                         $ units
        0, 857, 0,                  $ min, max, default data values
        ,,,,                        $ scaling, not used
        ,,,,                        $ link scaling, not used
        ;                           $ decimal places, not used

Param13 =
        0,                          $ first field shall equal 0
        ,,                          $ path size, path
        0x00,                       $ descriptor
        0xC8,                       $ data type: UDINT
        4,                          $ data size in bytes
        "Data Out A2", "",          $ name, help string
        "",                         $ units
        0, 857, 0,                  $ min, max, default data values
        ,,,,                        $ scaling, not used
        ,,,,                        $ link scaling, not used
        ;                           $ decimal places, not used

Param14 =
        0,                          $ first field shall equal 0
        ,,                          $ path size, path
        0x00,                       $ descriptor
        0xC8,                       $ data type: UDINT
        4,                          $ data size in bytes
        "Data Out B1", "",          $ name, help string
        "",                         $ units
        0, 857, 0,                  $ min, max, default data values
        ,,,,                        $ scaling, not used
        ,,,,                        $ link scaling, not used
        ;                           $ decimal places, not used

Param15 =
        0,                          $ first field shall equal 0
        ,,                          $ path size, path
        0x00,                       $ descriptor
        0xC8,                       $ data type: UDINT
        4,                          $ data size in bytes
        "Data Out B2", "",          $ name, help string
        "",                         $ units
        0, 857, 0,                  $ min, max, default data values
        ,,,,                        $ scaling, not used
        ,,,,                        $ link scaling, not used
        ;                           $ decimal places, not used

Param16 =
        0,                          $ first field shall equal 0
        ,,                          $ path size, path
        0x00,                       $ descriptor
        0xC8,                       $ data type: UDINT
        4,                          $ data size in bytes
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
            "Data Out C1", "",          $ name, help string
            "",                         $ units
            0, 857, 0,                  $ min, max, default data values
            ,,,,                        $ scaling, not used
            ,,,,                        $ link scaling, not used
            ;                           $ decimal places, not used

    Param17 =
            0,                          $ first field shall equal 0
            ,,                          $ path size, path
            0x00,                       $ descriptor
            0xC8,                       $ data type: UDINT
            4,                          $ data size in bytes
            "Data Out C2", "",          $ name, help string
            "",                         $ units
            0, 857, 0,                  $ min, max, default data values
            ,,,,                        $ scaling, not used
            ,,,,                        $ link scaling, not used
            ;                           $ decimal places, not used

    Param18 =
            0,                          $ first field shall equal 0
            ,,                          $ path size, path
            0x00,                       $ descriptor
            0xC8,                       $ data type: UDINT
            4,                          $ data size in bytes
            "Data Out D1", "",          $ name, help string
            "",                         $ units
            0, 857, 0,                  $ min, max, default data values
            ,,,,                        $ scaling, not used
            ,,,,                        $ link scaling, not used
            ;                           $ decimal places, not used

    Param19 =
            0,                          $ first field shall equal 0
            ,,                          $ path size, path
            0x00,                       $ descriptor
            0xC8,                       $ data type: UDINT
            4,                          $ data size in bytes
            "Data Out D2", "",          $ name, help string
            "",                         $ units
            0, 857, 0,                  $ min, max, default data values
            ,,,,                        $ scaling, not used
            ,,,,                        $ link scaling, not used
            ;                           $ decimal places, not used

[Assembly]
    Assem1 = "Input Assembly",          $ Name
             ,                          $ Path
             ,                          $ Data Size
             0x00,                      $ Descriptor
             ,,                         $ Reserved
             , BitStringVariant1;       $ Variable input

    BitStringVariant1 =
            "Output Data",,             $ Name, Help String
            ,,,                         $ Reserved
            Param1,                     $ Switch Selector
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
            0,Assem2,,                      $ Status/Fdbk Selection
            1,Assem3,Assem4,       $ DL A Out Selection or DL B Out Selection
            3,,Assem5,                      $ DL C Out Selection
            4,Assem6,;                      $ DL D Out Selection

    Assem2 = "Logic Status & Feedback",     $ Name
             ,                              $ Path
             4,                             $ Data Size
             0x00,                          $ Descriptor
             ,,                             $ Reserved
             16, Param10,                   $ Logic Status
             16, Param11;                   $ Feedback

    Assem3 = "DL A Out",                    $ Name
             ,                              $ Path
             8,                             $ Data Size
             0x00,                          $ Descriptor
             ,,                             $ Reserved
             32, Param12,                   $ Datalink A1 Out
             32, Param13;                   $ Datalink A2 Out

    Assem4 = "DL B Out",                    $ Name
             ,                              $ Path
             8,                             $ Data Size
             0x00,                          $ Descriptor
             ,,                             $ Reserved
             32, Param14,                   $ Datalink B1 Out
             32, Param15;                   $ Datalink B2 Out

    Assem5 = "DL C Out",                    $ Name
             ,                              $ Path
             8,                             $ Data Size
             0x00,                          $ Descriptor
             ,,                             $ Reserved
             32, Param16,                   $ Datalink C1 Out
             32, Param17;                   $ Datalink C2 Out

    Assem6 = "DL D Out",                    $ Name
             ,                              $ Path
             8,                             $ Data Size
             0x00,                          $ Descriptor
             ,,                             $ Reserved
             32, Param18,                   $ Datalink D1 Out
             32, Param19;                   $ Datalink D2 Out
```

In Figure 7-3.16 (above) when the value of Param1 is the default of 0x1F, the buffer of data described by BitStringVariant1 would be:

UINT Logic Status
INT Feedback
UDINT Data Out A1
UDINT Data Out A2
UDINT Data Out D1
UDINT Data Out D2

When the value of Param1 is 0x10, the buffer of data described by BitStringVariant1 would be:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

UDINT Data Out B1
UDINT Data Out B2
UDINT Data Out C1
UDINT Data Out C2
UDINT Data Out D1
UDINT Data Out D2

### 7-3.6.8.3 Array Entry Keyword

The "**Array**" keyword shall be used to describe a collection of elements of the same type of EDS Entry Keyword.

NOTE: If the elements of the array are not all the same size (for instance elements of type STRING could vary in length), then clients must parse the array data to access a specific index.

NOTE: Array of EPATHs cannot be supported because EPATH is a variable length type and there is no specified length for the EPATH.

The entry keyword shall consist of the character array, "**Array**", combined with a decimal number, e.g. "**Array1**".

**Table 7-3.40 ArrayN Entry Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Name | 1 | STRING | Required |
| Path | 2 | Padded EPATH | Optional |
| Descriptor | 3 | WORD | Optional |
| Help String | 4 | STRING | Optional |
| Reserved | 5,6,7 | Reserved | Not Used |
| Array Element Size | 8 | UINT | Optional |
| Array Element Type | 9 | Empty, AssemN, AssemExaN, ParamN, VariantN, BitStringVariantN, VariantExaN, ConstructedParamN | Required |
| Number of Dimensions | 10 | USINT | Required |
| Number of Dimension Elements | 11,12, 13 … | UDINT | Conditional [1] |

Table Footnotes:

1. One Number of Dimension Elements field is required for each dimension specified by field 10 (Number of Dimensions). Extra Number of Dimension Elements fields are not allowed.

#### 7-3.6.8.3.1 Name, Field 1

The "Name" field shall be a string giving a name to the array. This field may be used by the user interface.

#### 7-3.6.8.3.2 Path, Field 2

The second field, called "Path", identifies the address of the array within the device. If the value of the "Not Addressable" descriptor bit is 1 or this field is empty, the block described by this **ArrayN** entry is not directly addressable from the link.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.8.3.3 Descriptor, Field 3

The third field, "Descriptor", shall be a bit-field that describes certain properties of the Array. The bits of this field are described in Table 7-3.41. If this field is empty it shall be interpreted as 0.

The Descriptor shall be ignored if the Array element type is AssemN, the AssemN descriptor shall be used.

**Table 7-3.41 Descriptor Field Bit Value**

| Bit | Name | Value | Meaning |
|---|---|---|---|
| 0-3 | Reserved | 0 | |
| 4 | Not Addressable | 0 | If the Path field is not empty, the block described by this assembly entry is directly addressable from the link via the Set_Attribute and Get_Attribute services. |
| | | 1 | The block described by this assembly entry is not directly addressable from the link via the Set_Attibute or Get_Attribute services. |
| 5-7 | Reserved | 0 | |
| 8-10 | Write Allowed [1] | 0 | Indicates the array may or may not be able to be written when any I/O connections are open. |
| | | 1 | Indicates the array can be written when any I/O connections are open. |
| | | 2 | Indicates the array cannot be written when the I/O connection associated with this array is open. |
| | | 3 | Indicates the array cannot be written when any I/O connections are open. |
| | | 4-7 | Reserved |
| 11-15 | Reserved | 0 | |

Table Footnotes:

1- A write to the array can fail due to conditions other than I/O connections being open, like object state conflict.

#### 7-3.6.8.3.4 Help String, Field 4

The "Help String" field shall be a string providing help for the array. This optional field may be used by the user interface.

#### 7-3.6.8.3.5 Reserved Field 5

Field 5 shall be reserved for future definition and shall be empty.

#### 7-3.6.8.3.6 Reserved Field 6

Field 6 shall be reserved for future definition and shall be empty.

#### 7-3.6.8.3.7 Reserved Field 7

Field 7 shall be reserved for future definition and shall be empty.

#### 7-3.6.8.3.8 Array Element Size/Array Element Type, Fields 8 and 9

The "Array Element Size" field specifies the number of bits in each array element. The "Array Element Type" field specifies the type of each element in the array.

If the "Array Element Size" field is empty, the defined size of the corresponding "Array Element Type" field shall be used.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

NOTE: If the size of the Array Element Type can vary between elements, the Array Element Size field shall be empty.

The elements shall be placed into the data block least significant bit first. If the "Array Element Size" field is smaller than the defined size of the "Array Element Type" field, the least significant bits of the "Array Element Type" field shall be used. If the "Array Element Size" field is larger than the defined size of the "Array Element Type" field, the entire element shall be followed by zero pads to extend the element to the "Array Element Size". The data block represented shall be an integer number of bytes.

#### 7-3.6.8.3.9 Number of Dimensions, Field 10

The "Number of Dimensions" field specifies the number of dimensions in the array.

#### 7-3.6.8.3.10 Number of Dimension Elements, Fields 11, 12, 13…

The "Number of Dimension Elements" field specifies the number of elements for a dimension. The first "Number of Dimension Elements" field specifies the number of elements for the first dimension, the second for the second dimension, etc.

#### 7-3.6.8.3.11 Array Examples

Figure 7-3.17 shows the following examples of the "Array" keyword.

- A one dimensional array of fixed sized elements – see Array1 keyword
- A two dimensional array of fixed sized elements – see Array2 keyword
- A one dimensional array of variable sized elements – see Array3 keyword

**Figure 7-3.17 Array Keyword Example**

```
[Params]
   Param1 =
   0,            $ reserved, shall equal 0
   0,,           $ link path size, link path
   0x0000,       $ descriptor
   0xC7,         $ data type (UINT)
   2,            $ data size in bytes
   "Speed",      $ name
   "RPMs",       $ units
   "Element of speeds",   $ help string
   ,,100,        $ min, max, default data values
   ,,,,          $ mult, div, base, offset scaling
   ,,,,          $ mult, div, base, offset links
   ;             $ decimal places

   Param2 =
   0,            $ reserved, shall equal 0
   0,,           $ link path size, link path
   0x0000,       $ descriptor
   0xD9,         $ data type (SHORT_STRING)
   1,            $ data size in bytes
   "Device Name",         $ name
   "",           $ units
   "Device Name",         $ help string
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
   1,25,"Device Name n",  $ min, max, default data values
   ,,,,          $ mult, div, base, offset scaling
   ,,,,          $ mult, div, base, offset links
   ;             $ decimal places

[Assembly]
   Revision = 2;

   Array1 =                $ one dimension array
   "Channel 1 Speeds",     $ name of array
   "20 64 24 01 30 01",    $ link Path (vendor specific object attribute)
   0x0000,                 $ descriptor
   "Array of Speeds",      $ help string
   ,,,                     $ reserved
   16,                     $ array element size
   Param1,                 $ array element type
   1,                      $ one dimensional array
   5;                      $ number of elements in 1st dimension

   Array2 =                $ multiple dimension array
   "Channel Speeds",       $ name of array
   "20 64 24 01 30 02",    $ link Path (vendor specific object attribute)
   0x0000,                 $ descriptor
   "Array of Speeds",      $ help string
   ,,,                     $ reserved
   16,                     $ array element size
   Param1,                 $ array element type
   2,                      $ two dimensional array
   2,                      $ number of elements in 1st dimension
   3;                      $ number of elements in 2nd dimension

   Array3 = $ single dimension array of variable size elements (SHORT_STRING)
   "Device Names",         $ name of array
   "20 04 24 23 30 03",    $ link Path
   0x0000,                 $ descriptor
   "Array of Device Names",         $ help string
   ,,,                     $ reserved
   ,                       $ array element size (no size because element type
                           $ is variable size)
   Param2,                 $ array element type
   1,                      $ one dimensional array
   20;                     $ number of elements in array
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.6.8.4 Examples of Adapter/Rack Assem Data Formatting

The following examples illustrate the concepts of Adapter/Rack Assembly data formatting sections of the EDS file.

#### 7-3.6.8.4.1 Example 1

Example of a fixed slot size (2 bytes per slot), expanded adapter rack Assem, adapter excluded (Assem type = 1). The table below illustrates an example rack with an adapter in a 9-slot rack.

**Table 7-3.42 Rack Module Descriptions for Example 1**

| Slot 0 | Slot 1 | Slot 2 | Slot 3 | Slot 4 | Slot 5 | Slot 6 | Slot 7 | Slot 8 |
|---|---|---|---|---|---|---|---|---|
| Adapter | 16-bit input module | 16-bit input module | Empty slot | 16-bit output module | 16-bit input and output module | 16-bit input and output module | 16-bit output module | 16-bit input module |

The input data buffer being described by ProxyAssem1 and ProxiedAssem1 below is:

**Table 7-3.43 Slot Data Descriptions for Example 1**

| | |
|---|---|
| Slot 1 | 16 bits of data, format unspecified |
| Slot 2 | 4 bits of data described by Param1, 12 bits described by Param2 |
| Slot 3 | 16 bits of pad |
| Slot 4 | 16 bits of pad |
| Slot 5 | 4 bits of data described by Param1, 12 bits described by Param2 |
| Slot 6 | 4 bits of data described by Param1, 12 bits described by Param2 |
| Slot 7 | 16 bits of pad |
| Slot 8 | 4 bits of data described by Param1, 12 bits described by Param2 |

The following adapter and module EDS snippets could be used to describe the adapter rack input Assem for the example rack:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-3.18 EDS File Entries (partial) for Example 1**

Adapter EDS:

```
ProxyAssem1 =
  "Module Input Data",     $ Assem name
  "20 04 24 SLOT 30 0A",   $ no path
  2,                       $ 2 bytes of input per slot
  0b0000000000000010,      $ fixed slot size, expanded adapter rack
                           $ Assem, adapter excluded
  , ,                      $ reserved, reserved
  ModuleMemberList;        $ the module EDS specifies the format of
                           $ the module input data
```

Module in slot 1 EDS:

```
ProxiedAssem1 = ,,,,,,     $ Assem name, path, size, descriptor,
                           $ reserved, reserved
  16,;                     $ 16 bits of input data, format unspecified
```

Module in slot 2 EDS:

```
ProxiedAssem1 = ,,,,,,     $ Assem name, path, size, descriptor,
                           $ reserved, reserved
  4, Param1,               $ 4 bits of input data
  12, Param2;              $ 12 bits of input data
```

Module in slot 4 EDS:

```
ProxiedAssem1 = ,,,,,,     $ Assem name, path, size, descriptor,
                           $ reserved, reserved
  ;                        $ no input data
```

Module in slot 5 EDS:

```
ProxiedAssem1 = ,,,,,,     $ Assem name, path, size, descriptor,
                           $ reserved, reserved
  4, Param1,               $ 4 bits of input data
  12, Param2;              $ 12 bits of input data
```

Module in slot 6 EDS:

```
ProxiedAssem1 = ,,,,,,     $ Assem name, path, size, descriptor,
                           $ reserved, reserved
  4, Param1,               $ 4 bits of input data
  12, Param2;              $ 12 bits of input data
```

Module in slot 7 EDS:

```
ProxiedAssem1 = ,,,,,,     $ Assem name, path, size, descriptor,
                           $ reserved, reserved
  ;                        $ no input data
```

Module in slot 8 EDS:

```
ProxiedAssem1 = ,,,,,,     $ Assem name, path, size, descriptor,
                           $ reserved, reserved
  4, Param1,               $ 4 bits of input data
  12, Param2;              $ 12 bits of input data
```

### 7-3.6.8.4.2 Example 2

Example of a fixed slot size (2 bytes per slot), compressed adapter rack Assem, adapter excluded (Assem type = 2). The table below illustrates an example rack with an adapter in a 9-slot rack.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 7-3.44 Rack Module Descriptions for Example 2**

| Slot 0 | Slot 1 | Slot 2 | Slot 3 | Slot 4 | Slot 5 | Slot 6 | Slot 7 | Slot 8 |
|---|---|---|---|---|---|---|---|---|
| Adapter | 16-bit input module | 16-bit input module | Empty slot | 16-bit output module | 16-bit input and output module | 16-bit input and output module | 16-bit output module | 16-bit input module |

The input data buffer being described by ProxyAssem1 and ProxiedAssem1 below is:

**Table 7-3.45 Slot Data Descriptions for Example 2**

| | |
|---|---|
| Slot 1 | 16 bits of data, format unspecified |
| Slot 2 | 4 bits of data described by Param1, 12 bits described by Param2 |
| Slot 5 | 4 bits of data described by Param1, 12 bits described by Param2 |
| Slot 6 | 4 bits of data described by Param1, 12 bits described by Param2 |
| Slot 8 | 4 bits of data described by Param1, 12 bits described by Param2 |

The following adapter and module EDS snippets could be used to describe the adapter rack input Assem for the example rack:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-3.19 EDS File Entries (partial) for Example 2**

Adapter EDS:

```
ProxyAssem1 =
  "Module Input Data",      $ Assem name
  "20 04 24 SLOT 30 0A",    $ no path
  2,                        $ 2 bytes of input per slot
  0b0000000000000100,       $ fixed slot size, compressed adapter
                            $ rack Assem, adapter excluded
  ,,                        $ reserved, reserved
  ModuleMemberList;         $ the module EDS specifies the format of
                            $ the module input data
```

Module in slot 1 EDS:

```
ProxiedAssem1 = ,,,,,,      $ Assem name, path, size, descriptor,
                            $ reserved, reserved
  16,;                      $ 16 bits of input data, format unspecified
```

Module in slot 2 EDS:

```
ProxiedAssem1 = ,,,,,,      $ Assem name, path, size, descriptor,
                            $ reserved, reserved
  4, Param1,                $ 4 bits of input data
  12, Param2;               $ 12 bits of input data
```

Module in slot 5 EDS:

```
ProxiedAssem1 = ,,,,,,      $ Assem name, path, size, descriptor,
                            $ reserved, reserved
  4, Param1,                $ 4 bits of input data
  12, Param2;               $ 12 bits of input data
```

Module in slot 6 EDS:

```
ProxiedAssem1 = ,,,,,,      $ Assem name, path, size, descriptor,
                            $ reserved, reserved
  4, Param1,                $ 4 bits of input data
  12, Param2;               $ 12 bits of input data
```

Module in slot 8 EDS:

```
ProxiedAssem1 = ,,,,,,      $ Assem name, path, size, descriptor,
                            $ reserved, reserved
   4, Param1,               $ 4 bits of input data
   12, Param2;              $ 12 bits of input data
```

### 7-3.6.8.4.3 Example 3

Example of a variable sized, compressed adapter rack Assem, adapter excluded (Assem type = 3). The table below illustrates an example adapter rack with an adapter in a 9-slot rack.

**Table 7-3.46 Rack Module Descriptions for Example 3**

| Slot 0 | Slot 1 | Slot 2 | Slot 3 | Slot 4 | Slot 5 | Slot 6 | Slot 7 | Slot 8 |
|---|---|---|---|---|---|---|---|---|
| Adapter | 16-byte input module | 16-byte input module | Empty slot | 8-byte output module | 16-byte input and 4-byte output module | 16-byte input and 4 byte output module | 8-byte output module | 2-byte input module |

The input data buffer described by ProxyAssem1 and the ProxiedAssem1's is:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 7-3.47 Slot Data Descriptions for Example 3**

| | |
|---|---|
| Slot 1 | 128 bits of data, format unspecified |
| Slot 2 | 4 bits of data described by Param1, 12 bits described by Param2 |
| Slot 5 | 128 bits of data, format unspecified |
| Slot 6 | 128 bits of data, format unspecified |
| Slot 8 | 4 bits of data described by Param1, 12 bits described by Param2 |

The following adapter and module EDS snippets could be used to describe the adapter rack input Assem for the example rack:

**Figure 7-3.20 EDS File Entries (partial) for Example 3**

Adapter EDS:

```
ProxyAssem1 =
  “Module Input Data”,   $ Assem name
  “20 04 24 SLOT 30 0A”, $ no path
  ,                      $ size not specified, use member list size
  0b0000000000000110,    $ variable sized, compressed adapter rack
                         $ Assem, adapter excluded
  ,,                     $ reserved, reserved
  ModuleMemberList;      $ the module EDS specifies the format of
                         $ the module input data
```

Module in slot 1 EDS:

```
ProxiedAssem1 = ,,,,,,   $ Assem name, path, size, descriptor,
                         $ reserved, reserved
  128,;                  $ 128 bits of input data, format unspecified
```

Module in slot 2 EDS:

```
ProxiedAssem1 = ,,,,,,   $ Assem name, path, size, descriptor,
                         $ reserved, reserved
  4, Param1,             $ 4 bits of input data
  12, Param2;            $ 12 bits of input data
```

Module in slot 5 EDS:

```
ProxiedAssem1 = ,,,,,,   $ Assem name, path, size, descriptor,
                         $ reserved, reserved
  128;                   $ 128 bits of input data, format unspecified
```

Module in slot 6 EDS:

```
ProxiedAssem1 = ,,,,,,   $ Assem name, path, size, descriptor,
                         $ reserved, reserved
  128;                   $ 128 bits of input data, format unspecified
```

Module in slot 8 EDS:

```
ProxiedAssem1 = ,,,,,,   $ Assem name, path, size, descriptor,
                         $ reserved, reserved
  4, Param1,             $ 4 bits of input data
  12, Param2;            $ 12 bits of input data
```

### 7-3.6.8.5 Example of Indirect Parameter Handling

The following illustrates the Indirect Parameter Handling concept in the relevant sections of an EDS file.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This example represents a device with an input assembly (Assem1) which contains 3 members, of which the 3rd member can be selected by the user. The configuration assembly (Assem2) contains a single member, which is an indirect parameter, used to select the 3rd member of the input assembly. This example presumes the user accepts the default values for the parameters (Param1 and Param48). Param1 is the pointer and Param48 is the parameter selected to be in the input assembly.

**Figure 7-3.21 Indirect Parameter Handling Example**

```
[Params]
  Param1 =
     0,                      $ reserved, shall equal 0
     6,"20 0F 24 01 30 01",  $ Path Size, Path
     0x0400,                 $ Descriptor – Indirect Parameter
     0xC6,                   $ Data Type - USINT
     1,                      $ Data Size in bytes
     "Pointer",              $ name
     "",                     $ no units
     “Select parameter number to be used in input assembly", $ help string
     2,100,48,               $ min, max, default data values
     ,,,,                    $ mult, div, base, offset scaling
     ,,,,                    $ mult, div, base, offset links
     ;                       $ decimal places

  Param48 =
     0,                      $ reserved, shall equal 0
     6,"20 0F 24 30 30 01",  $ Path Size, Path
     0x0000,                 $ Descriptor
     0xC7,                   $ Data Type - UINT
     2,                      $ Data Size in bytes
     "Device Temperature",   $ name
     “Fahrenheit",           $ units
     “Device internal temperature",  $ help string
     ,,0x0055,               $ min, max, default data values
     ,,,,                    $ mult, div, base, offset scaling
     ,,,,                    $ mult, div, base, offset links
     ;                       $ decimal places

[Assembly]
  Assem1 =
     "Input",                $ name
     “20 04 24 02 30 30”,    $ Path
     ,                       $ Size - unspecified
     0x0000 ,                $ Descriptor – treat indirect parameters as pointers
     ,,                      $ Reserved
     ,Param3,      $ Member size, Member reference
     ,Param4,      $ Member size, Member reference
     ,Param1;      $ Member size, Member reference (user selected parameter)

  Assem2 =
     "Config",               $ name
     “20 04 24 01 30 30”,    $ Path
     ,             $ Size - unspecified
     0x0800 ,      $ Descriptor – treat indirect parameters as direct parameters
     ,,            $ Reserved
     ,Param1;      $ Member size, Member reference (user selectable
                   $ parameter for input assembly)
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Figure 7-3.21 lists portions of an EDS file used to illustrate the Assem(Exa) descriptor Indirect Parameter Handling.

The value of the third member of the Input data (Assem1) is 0x0055, because the value (and size) of the parameter pointed to is used (as specified by the Assem1 descriptor bits).

The value of the Config data (Assem2) is 48, because the value of the pointer parameter (and size) is used (as specified by the Assem2 descriptor bits).

### 7-3.6.9 Complete Electronic Data Sheet Example

An Electronic Data Sheet example is given below which is a compilation of the section examples given thus far in this chapter.

**Figure 7-3.22 Electronic Data Sheet Example**

```
[File]
   DescText = "Smart Widget EDS File";
   CreateDate = 04-03-1994;       $ created
   CreateTime = 17:51:44;
   ModDate = 04-06-1994;          $ last changed
   ModTime = 22:07:30;
   Revision = 2.1;                $ revision of EDS
[Device]
   VendCode = 65535;
   VendName = "Widget-Works, Inc.";
   ProdType = 43;
   ProdTypeStr = "Generic Device";
   ProdCode = 42;
   MajRev = 1;
   MinRev = 1;
   ProdName = "Smart-Widget";
   Catalog = "1499-DVG";
[ParamClass]
   MaxInst = 3;
   Descriptor = 0x0E;
   CfgAssembly = 3;

[Params]
   Param1 = 0, 6,"20 64 24 01 30 01", 0x0014, 0xC6, 1,"Preset","V",
             "User Manual p33", 0, 5, 1, 1, 1, 1, 0, , , , , 2;
   Param2 =                    $ parameter instance
            0,                 $ First field shall equal 0
             1, "20 04",       $ path size, path
             0x0014,           $ descriptor - in hex format
             0xC6,             $ data type: Unsigned 8-Bit Integer (USINT)
             1,                $ data size
             “Trigger”,        $ name
             “Hz”,             $ units
             “User Manual p49”, $ help string
             0, 2, 0,          $ min, max, default data values
             1, 1, 1, 0,       $ mult, div, base, offset scaling
             0, 0, 0, 0,       $ mult, div, base, offset links not used
             2;                $ decimal places

[Groups]
Group1 = "Setup", 2, 1, 2;         $ group 1
Group2 = "Monitor", 2, 2, 3;       $ group 2
Group3 = "Maintenance", 2, 1, 3;   $ group 3
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.6.10 Connection Manager Section

The Connection Manager section defines the set of target I/O connections supported by this device. These target I/O connections may exist as static connections predefined in the device or they may be dynamic I/O target connections which need to be configured in the device, e.g. as supported by the Connection Configuration object.

Target I/O connections defined in this section which do not specify the Safety real time transfer format are available on all device ports that support the Forward_Open service. Target I/O connections defined in this section, which specify the Safety real time transfer format, are available on all device ports that support the Forward_Open or Safety_Open service.

The Connection Manager section begins with the keyword [Connection Manager] and shall contain information concerning the number of types of application connections a device supports.

The following table identifies the entry keywords in the Connection Manager section that differ from the Common Object Class definition:

**Table 7-3.48 Connection Manager Section Entry Keyword Changes/Additions to the Common Object Class Entry Keywords (Table 7-3.5)**

| Entry Name | Entry Keyword | Required/Optional |
|---|---|---|
| Revision | Revision | Optional [2] |
| Maximum Instance Number | MaxInst | Optional [2] |
| Number of Static Instances | Number_Of_Static_Instances | Optional [2] |
| Maximum Number of Dynamic Instances | Max_Number_Of_Dynamic_Instances | Optional [2] |
| Connection | ConnectionN | Conditional [1] |
| Proxy Connection | ProxyConnectN | Conditional [1] |
| Proxied Connection | ProxiedConnectN | Conditional [1] |
| Connectivity Test Connection | ConnectivityTestConnect1 | Conditional [3] |
| Production Inhibit Time in Milliseconds Network Segment | PITNS | Conditional [4] |
| Production Inhibit Time in Microseconds Network Segment | PITNS_usec | Conditional [5] |
| Max Connection Establishment Time | MaxConnEstTime | Conditional [6] |

Table Footnotes:

1 A Connection entry shall be required for each non-explicit connection occurrence that can be described within an EDS file.

2 This keyword, although defined in section 7-3.6.1.3, Common Object Class Entry Keywords as Required, is Optional in the [Connection Manager] section.

3 Required if the Target Connection List object (see section 5B-6, Target Connection List Object) is implemented in the device, not allowed otherwise.

4 This keyword is required if the device supports the Production Inhibit Segment in Milliseconds, it is optional otherwise.

5 This keyword is required if the device supports the Production Inhibit Segment in Microseconds, it is optional otherwise.

6 This keyword is required if the Max Connection Establishment Time instance attribute (14) is supported, optional otherwise.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.6.10.1 Connection Entry Keywords

A Connection entry keyword shall represent all the options available for a given application path, therefore the Path field shall be unique. Using the Path as the key is necessary when a tool reads a connection configuration from a connection originator device for redisplay to a user.

There is one exception for the Path field being unique: A Connection entry keyword with application type Redundant Owner and a Connection entry keyword with application type Exclusive Owner can both specify the same Path field.

The Connection entry keyword shall be one of the following character arrays, "**Connection**", "**ProxyConnect**", "**ProxiedConnect**", combined with a number (decimal) or "**ConnectivityTestConnect1**". For example, "**Connection1**", "**ProxyConnect1**", or "**ProxiedConnect1**".

The decimal numbers shall start at 1 and increment for each additional "**Connection**" entry.

The decimal number shall not be required to start at 1 or increment for each additional "**ProxyConnect**" or "**ProxiedConnect**" entry.

If the device supports the Target Connection List object, the ConnectivityTestConnect1 keyword shall describe the Connectivity Test connection (see section 5B-6.3.1) specified by the Target Connection List object instance 1. This entry shall reflect all of the variations supported by the connection.

Commas shall separate all fields, and a semicolon shall indicate the end of the entry. A white space or nothing between commas shall be used for optional fields not provided.

The "**ProxyConnect**"and "**ProxiedConnect**" keywords are defined further in section 7-3.7 Modular EDS File Requirements. There are no provisions for defining a "Proxy/Proxied" Connectivity Test Connection for modular systems with a non-CIP backplane at this time.

Each entry shall contain the formatted fields shown in Table 7-3.49.

**Table 7-3.49 ConnectionN/ProxyConnectN/ProxiedConnectN/ConnectivityTestConnect1 Entry Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Trigger and transport | 1 | DWORD | Required |
| Connection parameters | 2 | DWORD | Required |
| O→T RPI | 3 | UDINT or ParamN | Optional |
| O→T size | 4 | UINT or ParamN | Conditional |
| O→T format | 5 | ParamN or AssemN or AssemExaN or ArrayN | Conditional |
| T→O RPI | 6 | UDINT or ParamN | Optional |
| T→O size | 7 | UINT or ParamN | Conditional |
| T→O format | 8 | ParamN or AssemN or AssemExaN or ArrayN | Conditional |
| Proxy Config size | 9 | UINT or ParamN | Optional |
| Proxy Config format | 10 | ParamN or AssemN or AssemExaN or ArrayN | Optional |
| Target Config size | 11 | UINT or ParamN | Optional |
| Target Config format | 12 | ParamN or AssemN or AssemExaN or ArrayN | Optional |
| Connection name string | 13 | STRING | Required |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Help string | 14 | STRING | Required |
| Path | 15 | Padded EPATH | Required |
| Safety ASYNC | 16 | See CIP Safety Spec (Volume 5, Chapter 5) | Conditional [1] |
| Safety Max Consumer Number | 17 | | Conditional [1] |

Table Footnotes:

1 - These fields are not allowed if the connection is not a safety connection. If the connection is a safety connection, see Volume 5.

#### 7-3.6.10.1.1 Trigger and transport mask

The bit assignments for the trigger and transport mask are as shown in Table 7-3.50

- A transport class bit shall be set to 1 for each class the connection supports. The other transport class bits shall be set to a 0.
- A trigger bit shall be set to a 1 for each trigger the connection supports. All other trigger bits shall be set to a 0.
- Only one of the application types shall be set to a 1. All other application type bits shall be set to a 0.
- For the direction bit: 0=client, 1=server, see Table 3-4.12 titled “Possible Values within Direction Bit”.

**Table 7-3.50 Trigger and transport mask bit assignments**

| Bit | Bit definition |
|---|---|
| 0 | transport class 0: null |
| 1 | transport class 1: duplicate detect |
| 2 | transport class 2: acknowledged |
| 3 | transport class 3: verified |
| 4 | transport class 4: non-blocking |
| 5 | transport class 5: non-blocking, fragmenting |
| 6 | transport class 6: multicast, fragmenting |
| 7-15 | transport class: reserved |
| 16 | trigger: cyclic |
| 17 | trigger: change of state |
| 18 | trigger: application |
| 19-23 | trigger: reserved |
| 24 | application type : listen-only |
| 25 | application type : input-only |
| 26 | application type : exclusive-owner |
| 27 | application type : redundant-owner |
| 28-30 | reserved |
| 31 | direction |

#### 7-3.6.10.1.2 Connection parameters

The bit assignments for the connection type and priority mask shall be as shown in Table 7-3.51. A bit shall be set to a 1 (on) for each connection type and priority the connection supports. All other bits shall be set to a 0 (off).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 7-3.51 Connection Parameters Bit Assignments**

| Bit | Bit definition |
|---|---|
| 0 | O→T fixed size supported |
| 1 | O→T variable size supported |
| 2 | T→O fixed size supported |
| 3 | T→O variable size supported |
| 4 – 5 | Obsolete, this capability now exists in AssemN descriptor bits 1-3 (Assem Type).<br>O→T number of bytes per slot in the O→T real time data packet for adapter rack connections.<br>0 = 1 byte<br>1 = 2 bytes<br>2 = 4 bytes<br>3 = 8 bytes |
| 6 – 7 | Obsolete, this capability now exists in AssemN descriptor bits 1-3 (Assem Type).<br>T→O number of bytes per slot in the T→O real time data packet for adapter rack connections.<br>0 = 1 bytes<br>1 = 2 bytes<br>2 = 4 bytes<br>3 = 8 bytes |
| 8 – 10 | O→T Real time transfer format.<br>0 = connection is pure data and is modeless.<br>1 = Use zero data length packet to indicate idle mode.<br>2 = reserved<br>3 = heartbeat<br>4 = 32-bit run/idle header<br>5 = Safety (See Volume 5)<br>6 thru 7 are reserved |
| 11 | reserved |
| 12 – 14 | T→O Real time transfer format<br>0 = connection is pure data and is modeless.<br>1 = Use zero data length packet to indicate idle mode<br>2 = reserved<br>3 = heartbeat.<br>4 = 32-bit run/idle header<br>5 = Safety (See Volume 5)<br>6 thru 7 are reserved |
| 15 | reserved |
| 16 | O→T connection type: NULL [1] |
| 17 | O-→ connection type: MULTICAST |
| 18 | O→T connection type: POINT2POINT |
| 19 | O→T connection type: reserved |
| 20 | T→O connection type: NULL [1] |
| 21 | T→O connection type: MULTICAST |
| 22 | T→O connection type: POINT2POINT |
| 23 | T→O connection type: reserved |
| 24 | O→T priority: LOW |
| 25 | O→T priority: HIGH |
| 26 | O→T priority: SCHEDULED |
| 27 | O→T priority: URGENT |
| 28 | T→O priority: LOW |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Bit | Bit definition |
|---|---|
| 29 | T→O priority: HIGH |
| 30 | T→O priority: SCHEDULED |
| 31 | T→O priority: URGENT |

Table Footnotes:

1 If both bit 16 and 20 are set (NULL Forward_Open) see section 7-3.6.10.2.13

#### 7-3.6.10.1.3 O→T RPI

The O→T RPI shall be the number of microseconds of the requested packet interval. The O→T RPI shall be a UDINT or a Param entry from the [Params] section that evaluates to a UDINT. If this field is empty, no constraints are placed on the O→T RPI.

#### 7-3.6.10.1.4 O→T size

The O→T size shall be the number of bytes sent in the O→T packet excluding the conditional sequence count (the sequence count does not exist for class 0 connections) and conditional 32-bit real time header (the real time transfer format is specified by bits 8-10 of the connection parameters field) sizes. The O→T size shall be a UINT or a Param entry that evaluates to a UINT. If this field is empty, the defined size of the O→T format shall be used.

#### 7-3.6.10.1.5 O→T format

The O→T format entry shall define the structure of the consumer buffer for this connection. Valid format descriptors shall be identifiers within the EDS file including

- a Param entry from the [Params] section;
- an Assem/AssemExa entry from the [Assembly] section.

This field may be empty indicating that the consuming format is not specified. This field shall not be empty if the O→T size field is empty. The O→T format shall not include the 32-bit real time header if it is present.

When the O→T format field is not empty and the O→T size field is not empty, if the O→T size field is smaller than the defined size of the O→T format, the least significant bytes of the O→T format fields shall be used. When the O→T format field is not empty and the O→T size field is not empty, if the O→T size field is larger than the defined size of the O→T format, the entire O→T format field shall be followed by zero pads to extend the O→T format to the "O→T Size".

#### 7-3.6.10.1.6 T→O RPI

The T→O RPI shall be the number of microseconds of the requested packet interval. The T→O RPI shall be a UDINT or a Param entry from the [Params] section that evaluates to a UDINT. If this field is empty, no constraints are placed on the T→O RPI.

#### 7-3.6.10.1.7 T→O size

The T→O size shall be the number of bytes sent in the T→O packet excluding conditional sequence count (the sequence count does not exist for class 0 connections) and conditional 32-bit real time header (the real time transfer format is specified by bits 12-14 of the connection parameters field) sizes. The T→O size shall be a UINT or a Param entry that evaluates to a UINT. If this field is empty, the defined size of the T→ format shall be used.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.10.1.8 T→ format

The T→O format shall define the structure of the producer buffer for this connection. Valid format descriptors shall be identifiers within the EDS file including

- a Param entry from the [Params] section;
- an Assem/AssemExa entry from the [Assembly] section.

This field may be empty indicating that the producing format is not specified. This field shall not be empty if the T→O size field is empty. The format shall include the 32-bit real time header if it is present.

When the T→O format field is not empty and the T→O size field is not empty, if the T→O size field is smaller than the defined size of the T→O format, the least significant bytes of the T→O format fields shall be used. When the T→O format field is not empty and the T→O size field is not empty, if the T→O size field is larger than the defined size of the T→O format, the entire T→O format field shall be followed by zero pads to extend the T→O format to the "T→ Size".

#### 7-3.6.10.1.9 Configuration

The Proxy Config size and Target Config size shall specify the size of the optional data segment that is appended to the path in the Forward_Open. The data segment shall be the concatenation of the two buffers described by the Proxy Config format and Target Config format. The sizes shall be the number of bytes and shall be a UINT or a Param entry from the [Params] section that evaluates to a UINT. If one of the config size fields is empty, the natural size of the corresponding config format field shall be used.

Valid config format fields shall be identifiers within the EDS file including

- a Param entry from the [Params] section;
- an Assem/AssemExa entry from the [Assembly] section.

The config format fields may be empty indicating that the config format is not specified. If both the config size and config format fields are empty, no data segment shall be appended to the path of the Forward_Open.

Target Config is used to describe the configuration data for the target device of the connection.

Proxy Config is used when there is a modular target (adapter and target module) to describe the configuration data the proxying adapter consumes.

The rules for the use of the Proxy Config and Target Config fields are described below.

For a non-modular device, the Target Config contains the configurable data for the device. The Proxy Config is not used.

For an adapter module's rack connection, the Proxy Config is meant to contain configuration data for the adapter related to the non-adapter modules in the adapter rack connection (e.g. external IDs of the non-adapter modules, fault and idle action settings, slot masks, etc.), the Target Config contains the collection of configuration data for the set of modules participating in the adapter rack connection.

For a non-adapter module direct connection, the Proxy Config is meant to contain configuration data for the adapter related to the non-adapter module (e.g. external ID of the non-adapter module, fault and idle action settings, etc.), while the Target Config is meant to contain configuration data for the non-adapter module itself.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

For an adapter module that supports a direct connection to its own data, rule 1 (the non-modular device) applies.

Table 7-3.52 shows how Modular/Non-Modular, Proxy/Proxied Connection and Modular Assembly are related to the Proxy Config and Target Config field rules.

**Table 7-3.52 Use of Proxy Config and Target Config**

| Device | Proxy/Proxied Connection Entry | Modular Assembly | Use of Proxy Config Data | Use of Target Config Data | Rule Number |
|---|---|---|---|---|---|
| Non-Modular | No | No | Not used. | Config data to be applied to the non-modular device. | 1 (Device) |
| Modular | No | Yes | Config data to be applied to the adapter module. | Config data to be applied to the non-adapter module(s). | 2 (Rack) |
| Modular | Yes | No | Config data to be applied to the adapter module. | Config data to be applied to the non-adapter module. | 3 (Direct) |
| Modular | No | No | Not used. | Config data to be applied to the adapter module. | 4 (Adapter) |

Table 7-3.53 and Table 7-3.54 further illustrate how the Proxy Config and the Target Config fields are used.

**Table 7-3.53 Non-Modular Device**

| Rule | Device |
|---|---|
| 1 (Device) | Target Config |

**Table 7-3.54 Modular Device**

| Rule | Adapter | Module 1 | Module 2 | … | Module N |
|---|---|---|---|---|---|
| 2 (Rack) | Rack related Proxy Config | Rack Member 1 Target Config | Rack Member 2 Target Config | … | Rack Member N Target Config |
| 3 (Direct) | Module related Proxy Config | | Target Config | … | |
| 4 (Adapter) | Target Config | | | … | |

### 7-3.6.10.1.10 Connection Name String

The connection name string shall be unique among the connections supported by the device. If a connection is available on more than one port, the same name shall be used in each of the [Connection ManagerN] section entries.

### 7-3.6.10.1.11 Help string

A tool may display the textual help character array. If no help string is to be provided a “null” string shall be used where a null string is defined as two double quotations: “” with no characters between the quotation marks.

### 7-3.6.10.1.12 Path

The path for a ConnectionN entry combined with Application Type Redundant Owner / Application Type Non-Redundant Owner shall be unique.

A path referencing the target object.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-3.23 [Connection Manager] Section Example**

```
[Params]
   Param1 =        $ specifies read buffer
   0, , ,          $ no path means not directly accessible
   0x0004,         $ descriptor : support scaling
   0xC6, 1,        $ USINT, 1 byte
   "Read",         $ name
   "", "",         $ units & help string
   64, 95, 64,     $ min, max, default data values
   1, 1, 1, -63,   $ mult, div, base, offset scaling
   0, 0, 0, 0, 0;  $ mult, div, base, offset link & decimal (not used)

   Param2 =        $ specifies write buffer
   0, , ,          $ no path means not directly accessible
   0x0004,         $ descriptor : support scaling
   0xC6, 1,        $ USINT, 1 byte
   "Write",        $ name
   "", "",         $ units & help string
   160, 191, 160,  $ min, max, default data values
   1, 1, 1, -159,  $ mult, div, base, offset scaling
   0, 0, 0, 0, 0;  $ mult, div, base, offset link & decimal $(not used)

[Connection Manager]
   Connection1 =
   0x04010002,     $ trigger & transport
                   $ class 1, cyclic, exclusive-owner
   0x44244401,     $ point/multicast & priority & real time format
                   $ fixed, 32-bit headers, scheduled,
                   $ O→T point-to-point, T→O multicast
   , 16, ,         $ O→T RPI, size, format
   , 12, ,         $ T→O RPI, size, format
   , ,             $ config part 1 (not used)
   , ,             $ config part 2 (not used)
   "read/write",   $ connection name
   "",             $ Help string
   "20 04 24 01 2C [Param2] 2C [Param1]";
```

### 7-3.6.10.1.13 Null Forward_Open Support

The Connection entry keyword shall be used to declare the types of Null Forward_Opens supported by the device (see section 3-5.4.2.2).

For the Null Forward_Open, matching connection (see section 3-5.4.2.2.2), the Connection entry that describes the I/O connection shall also describe the associated Null Forward_Open by having bits 16 and 20 set (NULL O→T and T→O Connection Type) in its Connection parameters DWORD.

For Null Forward_Open, no matching connection (see section 3-5.4.2.2.1), the following types exist:

1. Ping a device. A Connection entry shall be specified with the following field values:
   - Trigger and Transport – 0.
   - Connection Parameters – **bits 16 and 20 set all other bits 0**.
   - O→T RPI – empty.
   - O→T size – 0.
   - O→T format – empty.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- T→O RPI – empty.
- T→O size – 0.
- T→O format – empty.
- Proxy Config size – empty or 0.
- Proxy Config format – empty.
- Target Config size – empty or 0.
- Target Config format – empty.
- Connection name string – see section 7-3.6.10.2.10.
- Help string – see section 7-3.6.10.2.11.
- Path – "20 01 24 01".
- Safety ASYNC – not allowed.
- Safety Max Consumer Number – not allowed.

2. Configure a device. A Connection entry shall be specified with the following field values:

- Trigger and Transport – 0.
- Connection Parameters – bits 16 and 20 set all other bits 0.
- O→T RPI – empty.
- O→T size – 0.
- O→T format – empty.
- T→O RPI – empty.
- T→O size – 0.
- T→O format – empty.
- Proxy Config size – empty, UINT or Param.
- Proxy Config format – empty, Param or Assem or AssemExa.
- Target Config size – empty, UINT or Param.
- Target Config format – empty, Param or Assem or AssemExa.
- Connection name string – see section 7-3.6.10.2.10.
- Help string – see section 7-3.6.10.2.11.
- Path – see section 7-3.6.10.2.12.
- Safety ASYNC – not allowed.
- Safety Max Consumer Number – not allowed.

### 7-3.6.10.2 Production Inhibit Time in Milliseconds Network Segment Entry Keyword

The "PITNS" entry keyword shall contain the formatted field shown in Table 7-3.55. The "PITNS" entry keyword defines if the device supports the Product Inhibit Time in Milliseconds Network Segment. If this conditional keyword is omitted, this indicates the device does not support the Production Inhibit Time in Milliseconds Network Segment.

**Table 7-3.55 PITNS Keyword Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Supported | 1 | EDS_Field_Keyword, allowed values: Yes, No | Required |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.10.3 Production Inhibit Time in Microseconds Network Segment Entry Keyword

The “PITNS_usec” entry keyword shall contain the formatted field shown in Table 7-3.56. The “PITNS_usec” entry keyword defines if the device supports the Production Inhibit Time in Microseconds Network Segment. If this conditional keyword is omitted, this indicates the device does not support the Production Inhibit Time in Microseconds Network Segment.

**Table 7-3.56 PITNS_usec Keyword Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Supported | 1 | EDS_Field_Keyword, allowed values: Yes, No | Required |

#### 7-3.6.10.4 Max Connection Establishment Time Entry Keyword

The “MaxConnEstTime” entry keyword shall contain the formatted field shown in Table 7-3.57. The “MaxConnEstTime” entry keyword reports the same value as the Connection Manager object’s Max Connection Establishment Time attribute (14), see section 3-5.2.1.1.

**Table 7-3.57 MaxConnEstTime Keyword Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Processing Time | 1 | UINT | Required |

### 7-3.6.11 Port Specific Connection Manager Section

The Port Specific Connection Manager section defines the set of target I/O connections supported by this device on a specific port.

The Port Specific Connection Manager section begins with the keyword [Connection ManagerN], and contains information concerning the Connection Manager items the device supports for the port represented by the corresponding PortN entry.

The ConnectionN/ProxyConnectN/ProxiedConnectN and PITNS keywords, as defined in the [Connection Manager] section, are also defined for use in the [Connection ManagerN] section.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 7-3.58 Entry Keywords in the Connection ManagerN Class Section**

| Entry Name | Entry Keyword | Required/Optional |
|---|---|---|
| Connection | ConnectionN | Optional |
| Proxy Connection | ProxyConnectN | Optional |
| Proxied Connection | ProxiedConnectN | Optional |
| Production Inhibit Time in Milliseconds Network Segment | PITNS | Conditional [1] |
| Production Inhibit Time in Microseconds Network Segment | PITNS_usec | Conditional [2] |
| Max Connection Establishment Time | MaxConnEstTime | Conditional [3] |

Table Footnotes:

1 The "PITNS" entry keyword defines if the associated port of the device supports the Production Inhibit Time in Milliseconds Network Segment. If this conditional keyword is omitted, this indicates the associated port of the device does not support the Production Inhibit Time in Milliseconds Network Segment.

2 The "PITNS_usec" entry keyword defines if the associated port of the device supports the Production Inhibit Time in Microseconds Network Segment. If this conditional keyword is omitted, this indicates the associated port of the device does not support the Production Inhibit Time in Microseconds Network Segment.

3 This keyword is required if the the Max Connection Establishment Time instance attribute (14) is supported, optional otherwise.

A ConnectionN/ProxyConnectN/ProxiedConnectN entry in the [Connection ManagerN] section shall not specify the same Path field value as a ConnectionN/ProxyConnectN/ProxiedConnectionN entry in the [Connection Manager] section.

The N in ConnectionN/ProxyConnectN/ProxiedConnectN keywords in all [Connection Manager] and [Connection ManagerN] sections shall be unique.

**Figure 7-3.24 [Connection ManagerN] Section Example**

```
…
[Port]
 Port1 = …                        $ This device has a ControlNet port
 Port2 = …                        $ This is the device's first EtherNet/IP port
 Port3 = …                        $ This device has a DeviceNet port
 Port4 = …                        $ This is the device's second EtherNet/IP port
…
[Connection Manager]
 Connection1 = …      $ this connection applies to the ControlNet and
                      $ EtherNet/IP ports, this connection does not apply to
                      $ the DeviceNet port because the Forward_Open service is
                      $ not supported on DeviceNet

[Connection Manager1] $ connections in this section apply only to the port entry
                      $ specified by Port1
Connection2 = …       $ this connection only applies to the ControlNet port

[Connection Manager2] $ connections in this section apply only to the port entry
                      $ specified by Port2
Connection3 = …       $ this connection only applies to the first EtherNet/IP port

[Connection Manager4] $ connections in this section apply only to the port entry
                      $ specified by Port4
Connection4 = …       $ this connection only applies to the second EtherNet/IP
                      $ port
…
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.6.12 Port Section

The Port section begins with the keyword [Port] and shall describe the CIP ports available within a device. Every CIP routable port shall have a corresponding entry in this section. Non CIP routable ports may have a corresponding entry in this section, even if there is no corresponding Port Object instance implemented in the device. When no Port Object instance exists for the entry, the instance number and all fields within the entry shall be set as though the instance does exist.

The entry keyword for all ports shall consist of the character array "**Port**", combined with a decimal number corresponding to an instance of the port object. For example, Port1 is instance 1 of the Port Object. A white space or nothing between commas shall be used for optional fields not provided.

The following table identifies the entry keywords in the Port section that differ from the Common Object Class definition:

**Table 7-3.59 Port Section Entry Keyword Changes/Additions to the Common Object Class Entry Keywords (Table 7-3.5)**

| Entry Name | Entry Keyword | Required/Optional |
|---|---|---|
| Revision | Revision | Optional [2] |
| Maximum Instance Number | MaxInst | Optional [2] |
| Number of Static Instances | Number_Of_Static_Instances | Optional [2] |
| Maximum Number of Dynamic Instances | Max_Number_Of_Dynamic_Instances | Optional [2] |
| Port | PortN | Conditional [1] |

Table Footnotes

1 A Port entry is required for each CIP routable port occurrence in the device. Entries for ports that are not CIP routable are optional.

2 This keyword, although defined in section 7-3.6.1.2, Vendor Specific Object Class Sections

Optional Vendor Specific Object Class Sections shall be named **ClassN_VendorM** where N is the decimal vendor specific class code and M is the decimal Vendor ID. For instance, **[Class100_Vendor1534]** is the name of the class object section for class code 100, vendor 1534.

The Vendor Specific Object Class Sections may contain any of the Common Object Class Entry Keyword(s) described in section 7-3.6.1.3.

3 Common Object Class Entry Keywords as Required, is Optional in the [Port] section.

**Table 7-3.60 PortN Entry Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Port Type | 1 | EDS_Field_Keyword, see Table 7-3.61 for allowed values. | Required |
| Port Name | 2 | STRING | Required |
| Link Object | 3 | Padded EPATH | Required |
| Port Number | 4 | UINT | Required |
| Port Routing Capabilities | 5 | DWORD | Required |
| Associated Communication Objects | 6 | Array of Padded EPATH | Conditonal [1] |
| Port Specific | 7, 8, … | Port Specific | Port Specific |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|

Table Footnotes:

1. If the device supports more than one Port Object instance of the Port Type (field 1) of this PortN entry, this field is required, otherwise this field is optional.

The first field, called "**Port Type**", shall correspond to Port Object instance attribute 1, Port Type. The allowed Port Type field keywords are shown in the table below with corresponding Port object Port Type values:

**Table 7-3.61 Allowed Values for Port Type Field**

| Port Type Value | Corresponding Port Object Port Type (Instance Attribute 1) Value |
|---|---|
| <empty>[1] | 0 |
| <Reserved for legacy use>[2] | 1 |
| ControlNet | 2 |
| ControlNet_Redundant | 3 |
| TCP[3] | 4 |
| DeviceNet | 5 |
| <Reserved for legacy use>[2] | 6-99 |
| <Vendor Specific>[2] | 100-199 |
| CompoNet | 200 |
| ModbusTCP | 201 |
| ModbusSL | 202 |
| SERCOS_III | 203 |

Table Footnotes

1 When the Port Object Port Type instance attribute value is zero (0), the Port Type EDS field may either be empty or set to the Port Type Value assigned by this table to the supported technology. For example, an EtherNet/IP port that does not support routing may use 'TCP' for the Port Type even though its Port Object Port Type instance attribute is zero (0).

2 A vendor-specific field keyword (see section 7-3.3.2.1, EDS Content, for Vendor-Specific Keyword rules).

3 This shall indicate an EtherNet/IP capable TCP port.

The "**Port Name**" field corresponds to Port Object instance attribute 4 (Port Name), and shall have the same value as returned by that attribute.

The "**Link Object**" field corresponds to the Port Object instance attribute 3 (Link Object), and shall have the same value as returned by the Link Path portion of that attribute. This field shall be empty if there is no Link Object.

The "**Port Number**" field corresponds to the Port Object instance attribute 2 (Port Number), and shall have the same value as returned by that attribute.

The "**Port Routing Capabilities**" field corresponds to the Port Object instance attribute 10 (Port Routing Capabilities), and shall have the same value as returned by that attribute.

**Figure 7-3.25 [Port] Section Example**

```
[Port]
   Port1 =  DeviceNet,
   "Port A",              $ name of port
   "20 03 24 01",         $ instance one of the DeviceNet object
   2,                     $ port number 2
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
0x7F;                   $ port routing capabilities, full routing support

Port2 =  65535_Chassis,
"Chassis",              $ name of port
"20 9A 24 01",          $ vendor specific back-plane object
1,                      $ port number 1
0x3F;                   $ port routing capabilities, full routing support
```

### 7-3.6.13 Capacity Section

The Capacity section begins with the keyword [Capacity] and shall contain information relating to a device's communications capacity. Capacity section keywords are primarily oriented towards specifying a device's I/O communications capacity. The Capacity section also contains parameters not specifically related to I/O.

The entries in the Capacity section are intended to allow a software tool to walk through a set of configured connections for a system and determine whether each of the devices will be able to handle the configured load.

The following table shows the keywords defined for the Capacity section:

**Table 7-3.62 Capacity Section keywords**

| Entry Name | Entry Keyword | Number of Fields | Data Type | Required/Optional |
|---|---|---|---|---|
| Traffic Spec | TSpecN | 3 | n/a | Optional |
| Connection overhead | ConnOverhead | 1 | REAL | Optional |
| Maximum CIP connections | MaxCIPConnections | 1 | UINT | Conditional |
| Maximum I/O connections | MaxIOConnections | 1 | UINT | Conditional |
| Maximum explicit messaging connections | MaxMsgConnections | 1 | UINT | Conditional |
| Maximum I/O producers | MaxIOProducers | 1 | UINT | Conditional |
| Maximum I/O consumers | MaxIOConsumers | 1 | UINT | Conditional |
| Maximum I/O producer plus consumers | MaxIOProduceConsume | 1 | UINT | Conditional |
| Maximum I/O multicast producers | MaxIOMcastProducers | 1 | UINT | Conditional |
| Maximum I/O multicast consumers | MaxIOMcastConsumers | 1 | UINT | Conditional |
| Maximum consumers per multicast connection | MaxConsumersPerMcast | 1 | UINT | Conditional |

#### 7-3.6.13.1 Traffic Spec

The Traffic Spec represents the maximum packet rate (packets per second) across the range of implicit (I/O) connection sizes that the device supports. One or more TSpecN keywords shall be used to specify the points on a piece-wise linear curve as outlined below.

A TSpecN keyword is a point on the connection size vs. packets per second curve. Between any two adjacent points, the curve is assumed to be linear. Devices can specify as few or as many different points as needed to characterize their capacity at different connection sizes.

The following table shows the format of the TSpecN keyword:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 7-3.63 TSpecN Keyword Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| TxRx | 1 | EDS_Field_Keyword, possible values: Tx, Rx, and TxRx. | Required |
| ConnSize | 2 | UINT | Required |
| PacketsPerSecond | 3 | UDINT | Required |

- **TxRx** shall have one of the following values: "Tx", "Rx", or "TxRx". Tx shall indicate that the Traffic Spec applies to packets transmitted (produced) by the device. "Rx" shall indicate that the Traffic Spec applies to packets received (consumed) by the device. "TxRx" shall indicate that the Traffic Spec applies to both packets transmitted and received. If any of the Traffic Spec entries in a Capacity section use "TxRx" then ALL entries in the section shall use "TxRx".
- **ConnSize** shall be the CIP connection size, in bytes. That is, the ConnSize corresponds to the O→T and T→O connection size in the Forward Open service, or to the Produced_connection_size and Consumed_connection_size instance attributes of the Connection Object.
- **PacketsPerSecond** shall be the maximum number of implicit packets per second the device can process at the given ConnSize.

As an example, a device might have the following Traffic Specs:

- TSpec1 = TxRx, 2, 5000;
- TSpec2 = TxRx, 128, 4700;
- TSpec3 = TxRx, 256, 4000;
- TSpec4 = TxRx, 508, 2800;

In the example above, at a connection of size 2, the device can support 5000 packets per second. At a connection size of 128, the device can support 4700 packets per second. Linear interpolation is used to determine the capacity (packets per second) between those two sizes.

TSpecN entries shall be numbered consecutively, starting with N = 1. TSpecN entries shall be listed in order of increasing ConnSize. The first TSpec entry (TSpec1) shall use the smallest ConnSize supported by the device. The last TSpec entry shall use the largest ConnSize supported by the device.

Traffic Specs apply to a device's implicit communications capacity only. For EtherNet/IP and ControlNet devices, Traffic Specs shall apply to implicit connections that specify transport class 0 or transport class 1. For DeviceNet devices, Traffic Specs shall apply to implicit connections regardless of the transport class.

### 7-3.6.13.2 Generating TSpec Values

Each TSpec shows the device's maximum implicit packets per second at a given connection size. TSpecs should be generated over the range of connection sizes supported by the device. At each chosen connection size, the maximum packets per second should be determined using the smallest (fastest) RPI over the fewest number of connections possible. The recommended procedure to generate a Traffic Spec is outlined below:

- Establish an I/O connection to the device (or, from the device if it is an originator). Note: packets sent on the connection shall be a fixed size.
- Start with smallest RPI supported by the device.
- Measure the sustained packet rate (using a network analyzer, device diagnostics, or other means).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- Determine whether packets are being lost (using a network analyzer, device diagnostics, observing connection loss, or by other means).
- If there is packet loss, then increase the RPI (decrease packet rate) until no packets are being lost.
- If no packet loss, and if the RPI cannot be made any faster, then add connections until the device experiences packet loss or runs out of connections (RPIs for subsequent connections may need to vary).
- PacketsPerSecond for the TSpec shall be the measured packet rate. If the T→O and O→T (produced and consumed) connection sizes are not equal, then additional calculations must be done to generate the PacketsPerSecond number (see below).

Vendors should determine whether a device has the same capacity for Tx and Rx packets. If the device has a significant difference in Tx and Rx capacities, then it is desirable, though not required, to generate different Tx and Rx TSpecs. There are several options for generating separate Tx and Rx TSpecs:

1. using a null connection type in one direction
2. measuring Rx packet rates while holding the Tx packet size and rate constant (and vice versa for Tx packets)

Generating TSpec values can be difficult in that devices and the application software may support a limited number of connection sizes, and the connection sizes may be different in the T→O and O→T (produced and consumed) directions. For example, consider a device where the T→O connection size is 256, the O→T connection size is 2, and the maximum packet rate is measured at 4000 packets per second (2000 in each direction). Since different packet sizes are used, it is not possible to conclusively determine the maximum packet rates for the 2-byte and 256-byte sizes. We assume that at 2 bytes the device can process at least 4000 packets per second. At 256 bytes, we assume the device can process somewhere between 2000 and 4000 packets per second. In this situation, the recommended approach is to estimate the maximum packet rates at each of the two connection sizes, and then validate experimentally by setting up a number of connections and measuring packet rates or device CPU utilization if available.

It is acceptable for vendors to generate TSpecs in accordance with the device's limitations. For example, devices may support limited connection sizes and RPIs. If a device supports only a single fixed connection size then the device would have only one TSpec. If the device supports multiple connection sizes, then there shall be at least one TSpec at the minimum size, and one at the maximum size. Between the minimum and maximum, vendors should provide enough TSpecs to adequately characterize the I/O capacity of their devices.

After generating TSpec values, vendors should validate the TSpecs by setting up actual application connections over the range of supported connection sizes, RPI's, and number of connections to ensure that the device supports what the TSpecs state.

#### 7-3.6.13.3 Connection Overhead

The number of connections can also have an affect on a device's I/O capacity, independent of connection size and RPI. For example, due to connection management overhead, a device may be able to process 4000 packets with 2 connections, but only 3000 packets with 128 connections (given the same packet size).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

To account for connection overhead, the ConnOverhead keyword shall indicate a processing overhead to add per connection. ConnOverhead shall be expressed as a decimal number, representing the percentage of processing overhead for each connection (e.g., a value of .002 represents .2% overhead per connection). The value of ConnOverhead shall be less than 1.

Refer to the section on 7-3.6.13.6, Calculating Implicit Communications Usage for a description of how ConnOverhead is used.

The following shows an example of a ConnOverhead entry:

ConnOverhead = .002;

In the above example, for each connection there is a processing overhead of .002 (or .2%).

Devices are not required to include a ConnOverhead entry. Devices that do not incur additional overhead related to connection processing would typically not include a ConnOverhead entry.

### 7-3.6.13.4 Determining ConnOverhead Values

Values for the "ConnOverhead" parameters shall be determined by one of the following methods:

1. At a given packet rate, measure the device's CPU utilization (if available and accurate) at 1 connection and at the maximum number of connections supported by the device. ConnOverhead is the percent difference (expressed as a decimal number) in CPU utilization at 1 connection and at the maximum number of connections.
2. At a given connection size, measure the maximum packet rate at 1 connection (or at the smallest number of connections needed to generate max packet rate), and at the maximum connections supported by the device. ConnOverhead is the percent difference (expressed as a decimal number) in the maximum packet rate at 1 connection and at the maximum number of connections.

If possible, devices should determine their Traffic Specs using the smallest number of connections (ideally 1), and then determine their ConnOverhead using the methods outlined above. Otherwise, if the Traffic Specs are generated using a significant number of connections (relative to the maximum connections supported), then the connection-processing overhead will be accounted for in the Traffic Spec.

Some devices may have difficulty identifying connection overhead independent of their Traffic Specs. For example, a device that supports RPIs no faster than 50 ms may require a significant number of connections in order to generate a Traffic Spec. In this case, connection-processing overhead is already accounted for in the Traffic Spec. If the device also includes a ConnOverhead entry, the connection overhead would be included a second time.

If a device requires a significant number of connections (relative to the maximum number of connections supported) to generate its Traffic Specs, then the device should generally not include a ConnOverhead entry.

### 7-3.6.13.5 Capacity Assumptions

It is important that EDS capacity parameters be consistently interpreted and reported across different products. Values for capacity parameters – Traffic Spec rates, Connection Overhead, etc. – shall be assumed to indicate the device capacity that can be sustained under normal operating conditions in the absence of other (non-I/O) traffic, including:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- other CIP traffic (e.g., Class 3)
- non-CIP traffic (e.g., HTTP for EtherNet/IP devices, etc.)
- rejected multicast packets (due to limitations of Ethernet multicast filtering)

Future specification enhancements could address the effects of non-I/O traffic on I/O capacity.

### 7-3.6.13.6 Calculating Implicit Communications Usage

There are no requirements for particular behavior of a software tool that uses the capacity parameters. This section suggests an algorithm that could be used by a software tool to calculate and report device capacity.

A software tool would typically use the EDS capacity parameters in order to determine an application's projected communications usage. The tool would calculate, for each device in the application, the percentage of implicit communications capacity required by the configured connections. The tool could then report the percent utilization for each device, and whether any device has exceeded its capacity.

To calculate projected implicit communications usage, the software tool might do the following:

For each connection originator

For each configured implicit connection

Calculate the O→T (consumed) packet rate based on O→T RPI
Find the TSpec corresponding to the O→T size. If the connection size is variable, assume all packets are the largest size. Use linear interpolation if between TSpecs.
Determine the percentage of packet capacity used by O→T traffic and add to the sum of connection percentages for both originator and target

Calculate the T→O (produced) packet rate based on T→O RPI
Find the TSpec corresponding to the T→O size. If the connection size is variable, assume all packets are the largest size. Use linear interpolation if between TSpecs.
Determine the percentage of packet capacity used by T→O traffic and add to sum of connection percentages for both originator and target

For each connection originator and for each target

Multiply the ConnOverhead times the number of connections and add to the sum of connection percentages.

### 7-3.6.13.7 Connection Capacity Keywords

In addition to implicit communications capacity, it is useful for a device to provide information on the number of CIP connections supported, for both I/O and explicit communications. Connection capacity keywords are an important factor in allowing the user to determine whether the system as configured can successfully operate.

Characterizing a device's connection capacity is complicated by several factors:

- Depending upon implementation, multicast connections may share connection resources if they produce or consume the same data.
- CIP implementations may use different models for counting and limiting their connection resources


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

In order to account for different implementations, a number of connection capacity keywords are defined (refer to Table 7-3.64). As documented in Chapter 3, the term "connection" refers to the communications entity created using the Create service of the Connection Object, or the Forward_Open service of the Connection Manager Object.

The connection capacity keywords are further described below:

- **Maximum CIP connections (MaxCIPConnections)** – the maximum number CIP connections supported, including both I/O and explicit connections.
- **Maximum I/O connections (MaxIOConnections) –** the maximum number of I/O connections.
- **Maximum explicit connections (MaxMsgConnections) –** the maximum number of explicit message connections.
- **Maximum I/O producers (MaxIOProducers) –** the maximum number of I/O link producers.
- **Maximum I/O consumers (MaxIOConsumers) –** the maximum number of I/O link consumers.
- **Maximum I/O producers plus consumers (MaxIOProduceConsume) –** the maximum total of I/O link producers and consumers combined.
- **Maximum I/O multicast producers (MaxIOMcastProducers) –** the maximum number of multicast link producers.
- **Maximum I/O multicast consumers (MaxIOMcastConsumers) –** the maximum number of multicast link consumers.
- **Maximum consumers per multicast connection (MaxConsumersPerMcast) –** the maximum number of consumers allowed to join any one multicast connection.

The connection capacity keywords are conditional. The keywords may be used in particular combinations to model different approaches to counting and limiting connection resources. The allowable combinations of connection capacity keywords are show in the following table:

**Table 7-3.64 Allowable Combinations of Capacity Keywords**

| Allowed Combination of Connection Capacity Keywords | Description of Connection Counting Model | Additional Capacity Keywords Allowed (Optional) |
|---|---|---|
| MaxCIPConnections | The device counts CIP connections, with a common pool for both I/O and explicit message connections | MaxConsumersPerMcast |
| MaxIOConnections<br>MaxMsgConnections | The device counts CIP connections, with separate limits on I/O and explicit messaging connections. | MaxConsumersPerMcast |
| MaxIOProduceConsume<br>MaxMsgConnections | The device counts I/O and explicit message connections differently. For I/O connections the device counts producers and consumers. Each connection requires zero or one producer and zero or one consumer. Producers and consumers share a common pool. Multicast connections that request the same data are able to share producer and consumer resources.<br><br>For explicit messaging connections the device counts connections. | MaxIOMcastProducers<br>MaxIOMcastConsumers<br>MaxConsumersPerMcast |
| MaxIOProducers<br>MaxIOConsumers<br>MaxMsgConnections | Same as the above model, except I/O producers and consumers are maintained in separate pools. | MaxIOMcastProducers<br>MaxIOMcastConsumers<br>MaxConsumersPerMcast |

The connection capacity keywords are typically a result of limits built into a product's firmware or software. No measurement is generally needed to determine these limits, however it is expected that a product shall support the limits as stated in its EDS file.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.13.8 Considerations for Multiport Devices

Devices may have multiple ports, of the same type or different types. The keywords in the [Capacity] section shall apply device-wide, across all ports.

**Figure 7-3.26 Sample Electronic Data Sheet illustrating the Capacity Section**

```
[File]
   ...
[Device]
   ...
[Capacity]
   TSpec1 = TxRx, 2, 5000;      $ packets per sec @ 2 bytes
   TSpec2 = TxRx, 128, 4700;    $ packets per sec @ 128 bytes
   TSpec3 = TxRx, 256, 4200;    $ packets per sec @ 256 bytes
   TSpec4 = TxRx, 508, 3400;    $ packets per sec @ 508 bytes
   ConnOverhead = .002;         $ conn overhead
   MaxCIPConnections = 128;     $ no more than 128 total connections
   MaxConsumersPerMcast = 64;   $ 64 consumers per multicast connection
```

### 7-3.6.14 Connection Configuration Class Section

The Connection Configuration section defines the characteristics of the Connection Configuration Object (see Object Library, Chapter 5) implemented in this device, if a Connection Configuration Object implementation exists. The Common Object Class keywords may be used in the Connection Configuration section, see chapter 7-3.6.1. The following table identifies the entry keywords in the Connection Configuration Class section that differ from the Common Object Class definition. The Connection Configuration Class section begins with the keyword [Connection Configuration]. This section is required if the device supports the Connection Configuration object.

**Table 7-3.65 Connection Configuration Section Entry Keyword Changes/Additions to the Common Object Class Entry Keywords (Table 7-3.5)**

| Entry Name | Entry Keyword | Required/Optional |
|---|---|---|
| Number of Static Instances | Number_Of_Static_Instances | Optional [1] |
| Maximum Number of Dynamic Instances | Max_Number_Of_Dynamic_Instances | Optional [1] |
| Max Instances | MaxInst | Optional [1 3] |
| Mapping Attribute Format | MappingFormat | Conditional [2] |
| Originator to Target Image Sizes and Paths | O2TImages | Conditional [2] |
| Target to Originator Image Sizes and Paths | T2OImages | Conditional [2] |
| Connection Target Name | ConnectionTarget | Optional |
| Redundant Owner Connection | RedundantOwnerConnections | Optional |

Table Footnotes:

1 This keyword, although defined in section 7-3.6.1.3, Common Object Class Entry Keywords as Required, is Optional in the [Connection Configuration] section.

2 Required if the device's Connection Configuration Object is Revision 3 or higher, not allowed if Revision is 2 or lower.

3 The MaxInst keyword indicates the maximum number of Connection Configuration Object instances that can exist instead of the Common Object Class Entry definition which is the maximum instance number.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.14.1 Mapping Attribute Format Number Entry Keyword

The "**MappingFormat**" entry keyword shall contain the formatted field shown in Table 7-3.66.

**Table 7-3.66 MappingFormat Keyword Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Format | 1 | UINT | Required |

##### 7-3.6.14.1.1 Format, Field 1

The format number of the mapping attribute (class attribute 8) supported by this device's implementation of the Connection Configuration Object.

#### 7-3.6.14.2 Originator to Target Image Sizes and Paths Entry Keyword

The "**O2TImages**" entry keyword shall contain the formatted field shown in Table 7-3.67.
Table 7-3.67 O2TImages Keyword Format

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Reserved | 1, 2 | empty | |
| Image Size | 3, 5, 7 … | UINT | Field 3 is required. Fields 5, 7 … are conditional based on the existence of image tables. |
| Image Path | 4, 6, 8 … | Padded EPATH | Optional |

##### 7-3.6.14.2.1 Reserved, Field 1

Field 1 shall be reserved for future definition and shall be empty.

##### 7-3.6.14.2.2 Reserved, Field 2

Field 2 shall be reserved for future definition and shall be empty.

##### 7-3.6.14.2.3 Image Size

The number of 8 bit units in the Originator to Target image table. Field 3 specifies the size of Originator to Target image table 1; Field 5 specifies the size of Originator to Target image table 2; etc.

##### 7-3.6.14.2.4 Image Path

The Image Path is a logical or symbolic segment used to access the Originator to Target image table. Field 4 specifies the path used to access the Originator to Target image table 1; Field 6 specifies the path used to access the Originator to Target image table 2; etc.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.6.14.3 Target to Originator Image Sizes and Paths Entry Keyword

The "**T2OImages**" entry keyword shall contain the formatted field shown in Table 7-3.68.

**Table 7-3.68 T2OImages Keyword Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Reserved | 1,2 | empty | |
| Image Size | 3,5,7,… | UINT | Field 3 is required. Fields 5, 7… are conditional based on the existence of additional image tables. |
| Image Path | 4,6,8,… | Padded EPATH | Optional |

#### 7-3.6.14.3.1 Reserved, Field 1

Field 1 shall be reserved for future definition and shall be empty.

#### 7-3.6.14.3.2 Reserved, Field 2

Field 2 shall be reserved for future definition and shall be empty.

#### 7-3.6.14.3.3 Image Size

The number of 8 bit units in the Target to Originator image table. Field 3 specifies the size of Target to Originator image table 1; Field 5 specifies the size of Target to Originator image table 2; etc.

#### 7-3.6.14.3.4 Image Path

The Image Path is a logical or symbolic segment used to access the Target to Originator image table. Field 4 specifies the path used to access the Target to Originator image table 1; Field 6 specifies the path used to access the Target to Originator image table 2; etc.

### 7-3.6.14.4 Connection Target Name Entry Keyword

Each entry keyword shall be "**ConnectionTarget**" combined with a number (decimal), for example, "**ConnectionTarget1**". The number (decimal) shall be used to locate the corresponding "**Connection**" entry in the Connection Manager section of this device. One Connection Target Name entry may exist for each Connection entry specified in the Connection Manager section. If no ConnectionTarget entry corresponding to a Connection entry exists, the name from the Connection entry shall be used for the target entry name. The "**ConnectionTarget**" keyword shall contain the formatted field shown in Table 7-3.69.

**Table 7-3.69 ConnectionTarget Entry Keyword Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Name | 1 | STRING | Required |
| International Name | 2 | STRINGI | Optional |

#### 7-3.6.14.4.1 Name, Field 1

The name to display for the target entry that corresponds to the connection specified in the Connection Manager section by a Connection entry.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.14.4.2 International Name, Field 2

The connection target name expressed in STRINGI notation.

### 7-3.6.14.5 Redundant Owner Connection Support Entry Keyword

The "**RedundantOwnerConnections**" entry keyword is optional and shall contain the formatted field shown in Table 7-3.70. If this optional keyword is omitted, redundant connections cannot be originated.

**Table 7-3.70 RedundantOwnerConnections Keyword Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Supported | 1 | Field Keyword, possible values: Yes, No | Required |

#### 7-3.6.14.5.1 Supported, Field 1

This field indicates if redundant owner connections can be originated. If the field value is “Yes”, redundant connections can be originated. If the field value is “No”, redundant connections can not be originated. When redundant owner connections can be originated, as few as zero of the connections being originated, and as many as all the connections being originated may be redundant owner connections.

##### 7-3.6.14.5.1.1 Example Connection Configuration Section

The following is an example of a [Connection Configuration] section:

**Figure 7-3.27 Sample Electronic Data Sheet Illustrating the Connection Configuration Class Section**

```
$ The following EDS section describes a device that supports the 3rd
$ revision of Connection Configuration Object. The following are the
$ characteristics of this Connection Configuration Object implementation:
$
$  The maximum number of Connection Configuration Object instances
$   (connections or targets) that can be created is 128.
$  The Originator to Target Image (output) size is 4000 bytes long. The
$   Originator to Target Image can be read by performing a Get_Member
$   service to instance one of the Assembly Object (attribute 3).
$  The Target to Originator Image (input) size is 4000 bytes long. The
$   Target to Originator Image can be read by performing a Get_Member
$   service to instance two of the Assembly Object (attribute 3).
$  The name of a connection to this device is “Consume Data From”, the
$   connection path is “20 04 24 01 2C 01 2C 02”. If a target for a
$   connection to this device is created it shall be called “Send Data”.
$  This device does not support origination of Redundant Owner connections.
$  The Connection Configuration Object Open_Connection, Close_Connection
$   and Stop_Connection instance services are supported.
$  The ability to change this devices mode is supported by using the
$   Set_Attribute_Single service to the Connection Configuration Object
$   class attribute 0x33.

[File]
   ...
[Device]
   ...
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
[Connection Configuration]
   Revision = 3;         $ revision of the Connection Configuration object
                         $ supported in this device
   MappingFormat = 0;    $ Format of mapping attribute
                         $ (class attribute 8)
   MaxInst = 128;        $ maximum number of connection configuration
                         $ object instances supported
   O2TImages = ,,4000,”20 04 24 01 30 03”;
                         $ 4000 bytes of output image, path of output
                         $ image table
   T2OImages = ,,4000, “20 04 24 02 30 03”;
                         $ 4000 bytes of input image, path of input
                         $ image table
   ConnectionTarget1 = “Send Data”;
                         $ target name corresponding to Connection1
   RedundantOwnerConnections = No;
                         $ redundant output owner connections not
                         $ supported
   Number_Of_Static_Instances = 0;
   Max_Number_Of_Dynamic_Instances = 128;
   Instance_Services = 0x4C,    $ Open_Connection,
                       0x4D,    $ Close_Connection,
                       0x4E;    $ Stop_Connection,
                                $ services to instances supported
   Class_Attributes = 33;       $ Attribute 33 supported (set scanner mode)
   ...
[Assembly]
   Revision = 2;
   MaxInst = 177;
   Number_Of_Static_Instances = 14;
   Max_Number_Of_Dynamic_Instances = 0;
   Instance_Services = 0x18;         $ Get_Member service to instances
                                     $ supported (read of input and output
                                     $ image tables supported)
   ...
[Connection Manager]
   Connection1 =
   0x02010002,                       $ trigger and transport
   0x44244305,                       $ connection parameters
   ,0,,                              $ O→T RPI, size and format
   ,4,,                              $ T→O RPI, size and format
   ,,                                $ config part 1
   ,,                                $ config part 2
   “Consume Data From”,              $ connection name
   “”,                               $ help string
   “20 04 24 01 2C 01 2C 02”;        $ connection path
```

### 7-3.6.15 Event Enumeration Section

The Event Enumeration section [EventEnum] associates specific event or status codes within a device to an international string. The events identified may represent any event, error (including error response codes) or other condition within a device.

Events may be enumerated by the EventN entry keyword(s). There may be as many EventN entry keywords as necessary to identify all of the errors needing association.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 7-3.71 Fields in the EventN entry keyword**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Object Reference | 1 | Padded EPATH | Required |
| Reserved | 2 | empty | Not Used |
| Event Code 1 | 3 | UDINT | Required |
| Additional Code 1 | 4 | UDINT | Optional |
| Associated String 1 | 5 | STRINGI | Required |
| Event Code 2 | 6 | UDINT | Optional |
| Additional Code 2 | 7 | UDINT | Optional |
| Associated String 2 | 8 | STRINGI | Conditional |
| … | | | |
| Event Code J | (J-1)*3 +3 | UDINT | Optional |
| Additional Code J | (J-1)*3 +4 | UDINT | Optional |
| Associated String J | (J-1)*3 +5 | STRINGI | Conditional |

Where:

- Object Reference – This field shall only contain one of the following:
  - An EPATH identifying the object instance address (only Logical encoding of object class and explicit instance number)
  - An EPATH identifying all instances of a specific class (only Logical encoding of object class and no instance specified)
  - An empty entry

    This object reference is used to associate the following Error/Additional code(s) with the identified object(s).

    An empty entry specifies any instance of any object. If this field is empty, then (1) the Additional Code field shall also be empty and (2) the Event Code shall be within the non-object specific range.

    When this field is not empty it identifies some specific object instance (or object instances) within the device. The absence of an instance number specification indicates that the event enumerations pertain to any instances (including instance zero) of the related object.
- Reserved – This field shall be entered as an empty field. This field is reserved for future expansion of the EventN entry keyword.
- Event Code n – This field specifies an Event Code that is combined with the Object Reference and Additional Code fields to form a unique event identification.
- Additional Code n – This field specifies an additional (code qualification) value that is combined with the Object Reference and the Event Code fields to form a unique event identification. The presence of an empty field indicates any value is acceptable to form a value match.
- Associated String n – This international string is associated with the error identification. This field is required when Event Code n is specified.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The matching of an actual event condition (from an instance of a class and associated event and additional code) with a specific EventN entry keyword is dependant upon an application performing matching activities in the following order:

1. Actual error condition exactly matches an EventN entry keyword that explicitly specifies class and instance.
2. Actual error condition exactly matches an EventN entry keyword that explicitly specifies class and defaults to any instance (including instance zero).
3. Actual event condition's Event code matches an EventN entry keyword with an empty Object Reference (e.g. no class or instance specified) and exact match for Event code.

An example of an [EventEnum] section is shown below:

```
Event1 = “20 20 24 02”,   $ Object identification: Analog Input Group
                          $ Instance 2 only
   ,                       $ Reserved Field
                          $ ---
                          $ Error code 0x1F, Minor 0x00
   0x1F, 0x00,            $ Vendor specific error w/ additional code 0
   {2,
   {"eng",0xD0,4,"Message text 1" },
   {"deu",0xD0,4,"Anzeigetext 1" } $
   },

   0x1F, 0x05,            $ Vendor specific error w/ additional code 5
   { 2,
   {"eng",0xD0,4,"Message text 2" },
   {"deu",0xD0,4,"Anzeigetext 2" } $
   },

   0x0C, 0xFF,            $ Object state conflict translation
   { 2,
   {"eng",0xD0,4,"Message text 3"},
   {"deu",0xD0,4,"Anzeigetext 3"} $
   },

   0x10, 0x00,            $ Device state conflict translation
   { 2,
   {"eng",0xD0,4,"Message text 4"},
   {"deu",0xD0,4,"Anzeigetext 4"} $
   },

   0x10, 0x01,            $ Another device state conflict translation
   { 2,
   {"eng",0xD0,4,"Message text 5"},
   {"deu",0xD0,4,"Anzeigetext 5"} $
   };

Event2 = “20 01 24 03”,   $ Instance 3 of identity
   ,                      $ Reserved Field
   0x10, 0x04,
   { 2,
   {"eng",0xD0,4,"Subnet communication in progress - Try again later"},
   {"deu",0xD0,4,"Subnetzkommunikation läuft - Versuchen Sie es noch einmal
später"} $
   };
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
Event4 = “20 02”,          $ Message Router (any instance)
   ,                       $ Reserved Field
   0x10, 0x01,
   { 2,
   {"eng",0xD0,4,"EEPROM update in progress - Try again later"},
   {"deu",0xD0,4,"EEPROM Update läuft - Versuchen Sie es noch einmal
später"} $
   },

   0x10, 0x02,
   “{ 2,
   {"eng",0xD0,4,"Failed to initialize analog channel(s)"},
   {"deu",0xD0,4,"Analogkanal(-kanäle) konnte(n) nicht initialisiert
werden"} $
   },

   0x10, 0x04,
   { 2,
   {"eng",0xD0,4,"Remote communication channel active"},
   {"deu",0xD0,4,"Remotekommunikationskanal aktiv"} $
   };

Event5 = “”,                $ Any Object, Any instance
   ,                        $ Reserved Field
   0x1F, ,                  $ (Note – Null add code also)
   { 2,
   {"eng",0xD0,4,"Subnet devices unresponsive"},
   {"deu",0xD0,4,"Geräte im Subnetz antworten nicht"} $
   };
```

### 7-3.6.16 Symbolic Translation Section

The [SymbolicTranslation] section is used to publicize the translation between a Symbolic Segment or an ANSI Extended Symbol Segment encoded EPATH specification to the equivalent ArrayN, ParamN or AssemN entry keywords. Use of this section is required when a device utilizes a symbolic segment form of encoding within the EDS file.

The entry keyword for all symbolic translations shall consist of the character array, "Symbolic", combined with a decimal number, forming a unique keyword.

**Table 7-3.72 Fields in the Symbolic Keyword**

| Field Name | Field number | Data type | Required/Optional |
|---|---|---|---|
| Symbolic Definition | 1 | Padded EPATH | Required – Shall be a single symbolic segment specification |
| Translation | 2 | ParamN, AssemN, AssemExaN or ArrayN | Required – Shall be ParamN, AssemN, AssemExaN or ArrayN entry keyword reference |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

DeviceNet Specific Example:

```
[IO_Info]
   Default = 0x0001;                   $ Poll connection is default
   PollInfo = 0x0003, 1, 1;
   StrobeInfo = 0x0003, 2, 0;

   Input1 = 2, 16, 0x0001, “Specific volts”, 2, “61 41”;
   Input2 = 8, 64, 0x0002, “HMI Data”, 2, “61 42”;
   Output1 = 1, 8, 0x0001, “Motor Control”, 7, “66 4D 6F 74 6F 72 34”;

[SymbolicTranslation]
   Symbolic1 = “61 41”,                $ The symbol “A” translates to
      Param22;                         $ item described by Param22
   Symbolic2 = “61 42”,                $ The symbol “B” translates to
      Assem13;                         $ collection described by Assem13
   Symbolic3 = “66 4D 6F 74 6F 72 34 00”, $ The symbol “Motor4” is described
      Assem56;                            $ by Assem56 entry keyword

[Params]
   ...
   Param22 = ,
   6, “20 C6 24 03 30 14”,             $ Logical address within the device
   0, 0xC7, 2,                         $ UINT
   “Something vendor specific”,
   “Volts per meter”,                  $ Units
   “RF energy at equator”;
   ...

   Param8 = ,
   6, “20 19 24 01 30 03”,             $ Run1 attribute in Control
                                       $ Supervisor
   0, 0xC1, 1,                         $ BOOL
   “Run1”,                             $ Name
   “”,                                 $ Unitless
   “When true, Motor runs forward”;

   Param9 = ,
   6, “20 19 24 01 30 04”,             $ Run2 attribute in Control
                                       $ Supervisor
   0, 0xC1, 1,                         $ BOOL
   “Run2”,                             $ Name
   “”,                                 $ Unitless
   “When true, Motor runs in reverse”;
   ...

[Assembly]
   ...
   Assem13 = “HMI Specific Data”,
   ,                         $ No equivalent logical path
   8,                        $ Maximum size (8 bytes)
   0,                        $ Descriptor
   , ,                       $ Reserved fields
   16,,
   16,,
   16,,
   16, ;
   ...
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
   Assem56 = “Special Motor Control”,
   “20 04 24 CA 30 03”,      $ Accessible as Assembly 0xCA data attribute
   1,                        $ 1 byte
   0,                        $ Descriptor
   , ,                       $ Reserved fields
   1, Param8,                $ First bit is Run1
   1, Param9;                $ Second bit is Run2

Network Unspecific Example:
   ...


[Assembly]
   Assem1  = “Vendor Specific”,
   “20 04 24 01 30 03”,      $ In the device as assembly instance 1
   1,                        $ 1 byte assembly size
   0,                        $ Descriptor
   , ,                       $ Reserved fields
   4, “61 41”,               $ 4 bits from symbolic path “A”
   4, “61 42”;               $ 4 bits from symbolic path “B”
   ...

[SymbolicTranslation]
   Symbolic1 = “61 42”,      $ The symbol “B” translates to
   “20 C7 24 02 30 11”;      $ Object 199, Instance 2, Attribute 17

   Symbolic2 = “61 41”,      $ The symbol “A” translates to
   Param22;                  $ item described by Param22
   ...
```

### 7-3.6.17 Internationalization Section

The Internationalization section begins with the keyword [Internationalization] and shall describe EDS fields that can be internationalized. The table below shows the additional entries in the Internationalization section.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 7-3.73 EDS Entry Keywords in the Internationalization Section**

| Entry Name | Entry Keyword | Required/Optional |
|---|---|---|
| Descriptive Text | DescText | Optional |
| Device Type String | ProdTypeStr | Optional |
| Product Name | ProdName | Optional |
| Parameter | ParamN,<br>ProxyParamN, ProxiedParamN<br>ConstructedParamN<br>ProxyConstructedParamN<br>ProxiedConstructedParamN | Optional |
| Enumeration | EnumN | Optional |
| Group | GroupN | Optional |
| Assembly | AssemN, AssemExaN, ProxyAssemN, ProxyAssemExaN, ProxiedAssemN, ProxiedAssemExaN | Optional |
| Variant | VariantN, VariantExaN | Optional |
| Array | ArrayN | Optional |
| Connection | ConnectionN<br>ProxyConnectN<br>ProxiedConnectN | Optional |
| Connection Target | ConnectionTargetN | Optional |
| Port | PortN | Optional |
| Input | InputN, InputExaN | Conditional [1] |
| Output | OutputN, OutputExaN | Conditional [1] |

Table Footnotes:

1. This keyword is not allowed if the device is not a DeviceNet device. If the device is a DeviceNet device, see Volume 3.

### 7-3.6.17.1 Descriptive Text Entry Keyword

The “DescText” entry keyword is optional and shall contain the formatted field shown in Table 7-3.74.

**Table 7-3.74 DescText Keyword Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| International Text | 1 | STRINGI | Required |

#### 7-3.6.17.1.1 International Text, Field 1

A single line of international text displayed by the configuration tool. The EDS developer assigns a meaningful line of text for this entry. This field value overrides the “DescText” keyword field value in the [File] section.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.17.2 Device Type String Entry Keyword

The “ProdTypeStr” entry keyword is optional and shall contain the formatted field shown in Table 7-3.75.

**Table 7-3.75 ProdTypeStr Keyword Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| International Type | 1 | STRINGI | Required |

##### 7-3.6.17.2.1 International Type, Field 1

International text description of device type as shown in Chapter 6-7. The string for vendor specific device types is at the vendors’ discretion. This field value overrides the “ProdTypeStr” keyword field value in the [Device] section.

#### 7-3.6.17.3 Product Name Entry Keyword

The “ProdName” entry keyword is optional and shall contain the formatted field shown in Table 7-3.76.

**Table 7-3.76 ProdName Keyword Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| International Name | 1 | STRINGI | Required |

##### 7-3.6.17.3.1 International Name, Field 1

International text product name. Same as Identity Object, Attribute 13 (if Identity Object, Attribute 13 is defined). When displayed, truncation may occur to meet the display capabilities. This field value overrides the “ProdName” keyword field value in the [Device] section.

#### 7-3.6.17.4 Parameter Entry Keywords

The "ParamN", “ProxyParamN” and “ProxiedParamN” entry keywords are optional and shall contain the formatted fields shown in Table 7-3.77. These keywords correspond to the keywords of the same name in the [Params] section. When these keywords exist in the [Internationalization] section, fields 22, 23 and 24 for the corresponding ParamN, ProxyParamN or ProxiedParamN keyword in the [Params] section are overridden.

**Table 7-3.77 ParamN, ProxyParamN and ProxiedParamN Keywords Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| International Parameter Name | 1 | STRINGI | Required |
| International Engineering Units | 2 | STRINGI | Required |
| International Help String | 3 | STRINGI | Required |

##### 7-3.6.17.4.1 International Parameter Name, Field 1

The international textual parameter name. When displayed, truncation may occur to meet the display capabilities.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.17.4.2 International Engineering Units, Field 2

The international textual display units character array. If necessary, truncation of the retrieved text occurs to meet the maximum character array length allowed.

#### 7-3.6.17.4.3 International Help String, Field 3

The international textual help character array. If necessary, truncation of the retrieved text occurs to meet the maximum character array length allowed.

### 7-3.6.17.5 Constructed Parameter Entry Keywords

The "ConstructedParamN", “ProxyConstructedParamN” and “ProxiedConstructedParamN” entry keywords are optional and shall contain the formatted fields shown in Table Table 7-3.78. These keywords correspond to the keywords of the same name in the [Params] section.

**Table 7-3.78 ConstructedParamN, ProxyConstructedParamN and ProxiedConstructedParamN Keywords Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| International Name | 1 | STRINGI | Required |
| International Units String | 2 | STRINGI | Conditional [1] |
| International Member Unit String | 3 | {STRINGI, STRINGI…} [2] | Conditional [1] |
| International Help String | 4 | STRINGI | Required |
| International Member Help String | 5 | {STRINGI, STRINGI…} [2] | Conditional [1] |

Table Footnotes:

1. Required if the originating keyword contains this field, otherwise not allowed.
2. The number of STRINGIs is determined by the number of members in the Constructed Data Type.

#### 7-3.6.17.5.1 International Name, Field 1

The international textual name. When displayed, truncation may occur to meet the display capabilities.

#### 7-3.6.17.5.2 International Units String, Field 2

The international textual display units character array. If necessary, truncation of the retrieved text occurs to meet the maximum character array length allowed.

#### 7-3.6.17.5.3 International Member Units Strings, Field 3

The international textual member units character array. If necessary, truncation of the retrieved text occurs to meet the maximum character array length allowed.

#### 7-3.6.17.5.4 International Help String, Field 4

The international textual help character array. If necessary, truncation of the retrieved text occurs to meet the maximum character array length allowed.

#### 7-3.6.17.5.5 International Member Help Strings, Field 5

The international textual member help character array. If necessary, truncation of the retrieved text occurs to meet the maximum character array length allowed.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.17.6 Enumeration Entry Keyword

The "EnumN" entry keyword is optional and shall contain the formatted fields shown in Table 7-3.79.

If the tool supports the [Internationalization] section this “EnumN” keyword shall override the “EnumN” keyword in the [Params] section.

**Table 7-3.79 EnumN Keyword Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Value | 1,3,5,… | For integer data types, the data type of associated ParamN. For bit string data types, USINT | Field 1 Required, other fields optional |
| International Enumeration | 2,4,6,… | STRINGI | Required if field N-1 is specified, not allowed otherwise. |

##### 7-3.6.17.6.1 Value, Field 1, 3, 5,…

The value to enumerate.

##### 7-3.6.17.6.2 International Enumeration, Field 2, 4, 6, …

The international string that enumerates the value in the preceding field.

#### 7-3.6.17.7 Group Entry Keyword

The "GroupN" entry keyword is optional and shall contain the formatted field shown in Table 7-3.80.

**Table 7-3.80 DescText Keyword Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Internationalized Group Name String | 1 | STRINGI | Required |

##### 7-3.6.17.7.1 Internationalized Group Name String, Field 1

The international textual group name. This field value overrides the corresponding “GroupN” keyword name field value in the [Group] section.

#### 7-3.6.17.8 Assembly Entry Keywords

The "AssemN", "AssemExaN", “ProxyAssemN”, “ProxyAssemExaN”, “ProxiedAssemN” and “ProxiedAssemExaN” entry keywords are optional and shall contain the formatted fields shown in Table 7-3.81. These keywords correspond to the keywords of the same name in the [Assembly] section.

**Table 7-3.81 AssemN, AssemExaN, ProxyAssemN, ProxyAssemExaN, ProxiedAssemN and ProxiedAssemExaN Keywords Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Internationalized Assembly Name | 1 | STRINGI | Required |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.6.17.8.1 International Assembly Name, Field 1

The international textual assembly name. This field value overrides the corresponding "AssemN", "AssemExaN", “ProxyAssemN”, “ProxyAssemExaN”, “ProxiedAssemN” and “ProxiedAssemExaN” keyword name field value in the [Assembly] section.

### 7-3.6.17.9 Variant Entry Keywords

The "VariantN" and "VariantExaN" entry keywords are optional and shall contain the formatted fields shown in Table 7-3.82. These keywords correspond to the keywords of the same name in the [Assembly] section.

**Table 7-3.82 VariantN and VariantExaN Keywords Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Internationalized Variant Name | 1 | STRINGI | Required |
| Internationalized Variant Help String | 2 | STRINGI | Required |

#### 7-3.6.17.9.1 International Variant Name, Field 1

The international textual variant name. This field value overrides the corresponding "VariantN" and "VariantExaN" keyword name field value in the [Assembly] section.

#### 7-3.6.17.9.2 International Variant Help String, Field 2

The international textual variant help string. This field value overrides the corresponding "VariantN" and "VariantExaN" keyword help string field value in the [Assembly] section.

### 7-3.6.17.10 Array Entry Keywords

The "ArrayN" entry keyword is optional and shall contain the formatted fields shown in Table 7-3.77. This keyword corresponds to the keyword of the same name in the [Assembly] section.

**Table 7-3.83 ArrayN Keyword Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Internationalized Array Name | 1 | STRINGI | Required |
| Internationalized Array Help String | 2 | STRINGI | Required |

#### 7-3.6.17.10.1 International Array Name, Field 1

The international textual array name. This field value overrides the corresponding "ArrayN" keyword name field value in the [Assembly] section.

#### 7-3.6.17.10.2 International Array Help String, Field 2

The international textual array help string. This field value overrides the corresponding "ArrayN" keyword help string field value in the [Assembly] section.

### 7-3.6.17.11 Connection Entry Keyword

The "ConnectionN", "ProxyConnectN", "ProxiedConnectN" and “ConnectivityTestConnect1” entry keywords are optional and shall contain the formatted fields shown in Table 7-3.84.

The "ConnectionN", "ProxyConnectN", "ProxiedConnectN" and “ConnectivityTestConnect1” keywords correspond to the keywords of the same name in the [Connection Manager] section.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 7-3.84 ConnectionN, ProxyConnectN, ProxiedConnectN and "ConnectivityTestConnect1" Keyword Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Internationalized Connection Name | 1 | STRINGI | Required |
| Internationalized Connection Help String | 2 | STRINGI | Required |

#### 7-3.6.17.11.1 International Connection Name, Field 1

The international textual connection name. This field value overrides the corresponding "ConnectionN", "ProxyConnectN", "ProxiedConnectN" and "ConnectivityTestConnect1" keyword name field values in the [Connection Manager] section.

#### 7-3.6.17.11.2 International Connection Help String, Field 2

The international textual connection help string. This field value overrides the corresponding "ConnectionN", "ProxyConnectN", "ProxiedConnectN" and "ConnectivityTestConnect1" keyword help string field values in the [Connection Manager] section.

### 7-3.6.17.12 Port Entry Keyword

The "PortN" entry keyword is optional and shall contain the formatted field shown in Table 7-3.85.

**Table 7-3.85 PortN Keyword Format**

| Field name | Field Number | Data type | Required/Optional |
|---|---|---|---|
| Internationalized Port Name | 1 | STRINGI | Required |

#### 7-3.6.17.12.1 International Port Name, Field 1

The international textual port name. This field value overrides the corresponding "PortN" keyword name field value in the [Port] section.

## 7-3.7 Modular EDS File Requirements

This section describes the concept and contents of a Modular EDS and specifies the usage requirements.

### 7-3.7.1 Modular Section

The [Modular] section begins with the keyword [Modular] and shall describe a chassis based system. The two types of modular devices shall be:

- Chassis;
- Module.

A [Modular] section that describes a chassis shall contain a required keyword "**DefineSlotsInRack**" as shown in Figure 7-3.28. The single field on this entry shall be a 16-bit unsigned integer (UINT) indicating the number of slots in the chassis. Even though an electronic key is defined for the chassis, it need not be addressable from the link. The SLOT keyword used in path definitions in the [Connection Manager] section shall range from 0 to the number of slots minus 1. The keyword "**SlotDisplayRule**" is optional. The single field on this entry shall be a parameter from the [Params] section (ParamN), which defines the translation between internal and external slot number.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-3.28 [Modular] section describing a chassis**

```
[File]
   DescText = "Wonder Chassis EDS file";
   CreateDate = 09-01-1997;
   CreateTime = 17:23:00;
   Revision = 1.1;
[Device]
   VendCode = 65535;
   VendName = "Widget Works, Inc.";
   ProdType = 101;
   ProdTypeStr = "Widget Works Generic";
   ProdCode = 1;
   MajRev = 1;
   MinRev = 1;
   ProdName = "Widget Chassis";
   Catalog = "1234-chassis";
[Params]
   Param1 =
   0,                              $ first field shall equal 0
   ,,                              $ path size, path
   0x0006,                         $ descriptor
   0xC6,                           $ data type: Unsigned 8-bit Integer
   1,                              $ data size in bytes
   "Slot Naming Convention",       $ name
   "",                             $ units
   "",                             $ help string
   0,4,0,                          $ min, max, default data values
   0,0,0,0,                        $ mult, dev, base, offset scaling
   0,0,0,0,                        $ mult, dev, base, offset link not used
   0;                              $ decimal places not used
   Enum1 =
   0,"n/a",1,"0",2,"1",3,"2",4,"3";

 [Modular]
   DefineSlotsInRack = 5;
   SlotDisplayRule = Param1;
```

A [Modular] section that describes a module shall contain the "Width" and "Rack" entries.

The required entry with the keyword "**Width**" shall have two fields. The first field is required and indicates how many slots of the chassis are consumed by the module. The field shall be a 16-bit unsigned integer (UINT). The second field is optional and indicates which slot of the module connects to the chassis. If this optional field is omitted, slot 0 of the module is the connecting slot. The field shall be a 16-bit unsigned integer (UINT).

The entry keyword for all chassis, into which the module can be placed, shall consist of the character array, "**Rack**", combined with a decimal number. The numbers shall start at 1 for the first chassis, and shall be incremented for each additional chassis. Commas shall separate all fields, and a semicolon shall indicate the end of the entry. The fields for the "Rack" entries shall be as shown in Table 7-3.86.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 7-3.86 Field for the Rack entry keyword**

| Field name | Field number | Data type | Required/Optional |
|---|---|---|---|
| Vendor ID | 1 | UINT | Required |
| Product Type | 2 | UINT | Required |
| Product Code | 3 | UINT | Required |
| Major Revision | 4 | USINT | Required |
| Minor Revision | 5 | USINT | Required |
| Reserved | 6, 7, 8 | empty | Not Used |
| Legal Slot | 9, 10, 11 … | UINT | Required |

The "Vendor ID", "Product Type", "Product Code", "Major Revision" and "Minor Revision" field shall identify the electronic key of the chassis into which the module may be placed. The reserved field shall be empty. The "Legal Slot" fields shall identify the slots into which the module may be placed. The EDS for the module shall contain one "Rack" entry for each chassis into which the module may be placed as shown in Figure 7-3.29.

**Figure 7-3.29 [Modular] Section Describing a Module With a Network Connection**

```
[Modular]
   Width = 1;
   Rack1 =                            $ this module can only plug into
   65535, 101, 1, 1, 1,,,,            $  slot 0 of this five slot chassis
   0;

   Rack2 = 65535, 101, 2, 1, 1,,,,
   0;
```

## 7-3.7.2 Modular Additions to Basic EDS Sections

### 7-3.7.2.1 Path format convention addition for Modular EDS Files – (EPATH)

An EPATH in a Modular EDS file may also contain the other following references (in addition to the references specified in the EPATH definition in section 7-3.5.1.5):

- the keyword SLOT;
- the keyword SLOT_MINUS_ONE;

The keyword, SLOT, shall always evaluate to a USINT. The values substituted for the SLOT keyword shall correspond to the position of a module in a back plane. The keyword, SLOT_MINUS_ONE, shall always evaluate to a USINT. The values substituted for the SLOT_MINUS_ONE keyword shall correspond to the position of a module in the back plane minus 1.

EPATH example with "SLOT": “20 04 24 SLOT 30 03”

If the module position is slot 4 the CIP resulting EPATH would be “20 04 24 04 30 03”.

EPATH example with "SLOT_MINUS_ONE": “20 04 24 SLOT_MINUS_ONE 30 03”

If the module position is slot 4 the CIP resulting EPATH would be “20 04 24 03 30 03”.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.7.2.2 Additions to the Modular Section

A [Modular] section that describes a module that does not have a link addressable Identity object may contain the entry keyword "**ExternalID**". The keyword shall have a single field of data type EDS_External_Id. This field identifies the module. Modules that may be placed into slot 0 shall have an addressable Identity object and shall have no "**ExternalID**" entry.

A [Modular] section that describes a module that has a link connection and can be placed in slot 0 may contain the keyword "**GenericID**". The keyword shall have a single value of data type EDS_External_Id. This field shall be included in the data segment for a module connection in place of the ExternalID when no module keying is desired.

A [Modular] section that describes a module that has a link connection and can be placed in slot 0 may contain the keyword "**ExternalIDExactMatch**". The keyword shall have a single value, Yes or No. Yes shall indicate that the ExternalID specifies one specific device, No shall indicate that the ExternalID specifies one of a set of compatible devices. If the "**ExternalIDExactMatch**" does not exist the default condition shall be that the ExternalID specifies one specific device.

A [Modular] section that describes a module that has a link connection and can be placed in slot 0 may contain the entry keyword "**Query**". This keyword shall have 4 fields as described in Table 7-3.87.

**Table 7-3.87 Query Entry Format**

| Field Name | Field Number | Data Type | Required/Optional |
|---|---|---|---|
| Query Path | 1 | Packed EPATH | Required |
| Query Service | 2 | USINT | Required |
| Module ID Length | 3 | USINT | Required |
| Empty Slot ID | 4 | EDS_External_Id | Required |

- Query Path – the service path to be sent with the Query service to the module described by this EDS.
- Query Service – the service sent with the Query Path, sent to the module described by this EDS (i.e. 0x01 – get attribute all or 0x0E – get attribute single).
- Module ID Length – the length, in bytes, of a module ID returned in response to the Query service. This value shall be in the range of 1 to 16.
- Empty Slot ID – the ID returned for an empty slot in the chassis. The length of the Empty Slot ID shall be the Module ID Length.

The response to the Query service is an array of octets. The Module ID Length identifies the number of octets returned for each slot in the module's (described by this EDS) chassis. The Module ID's are returned in slot order, one Module ID for each slot in the chassis, starting with slot 1. For empty slots the Empty Slot ID is returned.

If a multiple slot module is in the chassis, the external identifier for the module shall appear one time for each slot occupied in the array returned from a query. A query shall only be addressed to a module in slot 0.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-3.30 [Modular] Section Describing a Module With a Slot 0 Link connection**

```
[Modular]
  Width = 1;
  Rack1 =                  $ this module can plug into
  65535, 101, 1, 1, 1,,,, $ slot 0
  0;                       $ this five slot chassis

  Rack2 =
  65535, 101, 2, 1, 1,,,,
  0;

  Query = “20 04 24 07 30 03”,1,2,”FF FF”;
  GenericID = “00 00”;
  ExternalIDExactMatch = No;
```

**Figure 7-3.31 [Modular] Section Describing a Module Without an Identity Object**

```
[Modular]
  Width = 1;
  Rack1 =                      $ this module can plug into
  65535, 101, 1, 1, 1,,,,      $  slots 1, 2, 3 and 4 of
  1, 2, 3, 4;                  $  this five slot chassis

  Rack2 =
  65535, 101, 2, 1, 1,,,,
  1, 2, 3, 4, 5, 6, 7;

  ExternalID = “12 34”;
```

### 7-3.7.2.3 Additions to the Parameter Section

The “**ProxyParam”, “ProxiedParam”, ”ProxyConstructedParam” and ”ProxiedConstructedParam”** keywords shall be used to describe parameters that are proxied by a CIP adapter device to another device that does not support the CIP protocol. An example of this is a ControlNet adapter module (the device proxying the connection) in a multiple slot I/O rack with an analog I/O module (the device the connection is proxied for).

In the following sections wherever “**ProxyParam**” is used “**ProxyConstructedParam**” may also be used. The same is true for wherever “**ProxiedParam**” is used “**ProxiedConstructedParam**” may also be used.

The “**ProxyParam”** shall exist in the EDS for the device that performs the proxy.

The “**ProxiedParam”** keyword shall exist in the EDS for the device that the proxy is performed for.

The information in the [Modular] section shall be used to associate EDS files containing “**ProxyParam”** keywords to EDS files containing “**ProxiedParam**” keywords. This association shall exist when both EDS files specify a matching Rack entry. “ProxyParam”/”ProxiedParam” and “Param” number ranges are independent of each other.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The decimal number (that is combined with "**ProxyParam**" and "**ProxiedParam**") shall be used to match a "**ProxyParam**" to a "**ProxiedParam**". The field values of a matched "**ProxyParam"** and "**ProxiedParam"** pair shall be combined to constitute the same field value information that exists in a single "**Param**" entry. This combination shall be done by using the field value from the "**ProxyParam"** unless that field value is the keyword "**Module**". When the field value specified in the "**ProxyParam"** is "**Module"** the field value specified in the "**ProxiedParam"** shall be used. It shall be legal to specify field values for "**ProxiedParam"** entries whose corresponding field value in the "**ProxyParam"** is not "**Module"**, however, these field values shall not be used, they shall exist only for documentation.

Another keyword may also exist in the [Params] section. This keyword shall be used to provide minimum, maximum and default values to be added to the "**ProxyParam**" minimum, maximum and default values. This entry keyword shall be "**ProxyParamSizeAdder**", combined with the decimal number from the corresponding "**ProxyParam"** entry. Commas shall separate all fields, and a semicolon shall end the entry. Each "**ProxyParam"** entry shall consist of a Minimum Value, Maximum Value and Default Value fields. The definition of these fields matches the "**Param**" definitions. The "**ProxyParamSizeAdder**" keyword provides a means for an adapter on a module connection ("**ProxyConnect**") to add adapter data to the module data and return the combined data on the connection.

Another keyword may also exist in the [Params] section that corresponds to the "**ProxyParam**", "**ProxyEnum**". "**ProxyEnum"** has the same definition as "**Enum**" except it is associated with "**ProxyParam**" instead of "**Param**". A second keyword may also exist in the [Params] section that corresponds to the "**ProxiedParam", "ProxiedEnum". "ProxiedEnum"** has the same definition as "**Enum**" except it is associated with "**ProxiedParam**" instead of "**Param**".

A ProxyEnum keyword may exist without a corresponding ProxiedEnum keyword. The value(s) enumerated by the ProxyEnum shall be used for the corresponding ProxyParam/ProxiedParam.

A ProxiedEnum keyword may exist without a corresponding ProxyEnum keyword. The value(s) enumerated by the ProxiedEnum shall be used for the corresponding ProxyParam/ProxiedParam.

A ProxyEnum keyword may exist with a corresponding ProxiedEnum keyword. The value(s) enumerated by the ProxyEnum shall be ignored and the value(s) enumerated by the ProxiedEnum ¬shall be used for the corresponding ProxyParam/ProxiedParam.

For bit fields specified in proxy keyword fields, the binary value shall allow a character of M (in addition to the 0 and 1 characters). The character M shall indicate to use the character specified in the proxied keyword field.

For example, the descriptor bits for a ProxyAssemN whose value is 0b0000000000000001M shall indicate that the assembly is a fixed sized expanded rack assembly and the ProxiedAssemN defines the allow value edit setting.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

#### 7-3.7.2.4 Additions to the Assembly Section

The following entries are additional values allowed for the "Member Reference" field within the Assembly section:

- ExternalID;
- InputSlotMask0;
- InputSlotMask1;
- OutputSlotMask0;
- OutputSlotMask1;
- ConfigSlotMask0;
- ConfigSlotMask1.

The "**ExternalID**" keyword shall indicate the usage of the device "**ExternalID**" value in the case when device keying is desired or the "**GenericID**" value in the case when module keying is not desired.

The "**InputSlotMask0**" keyword shall indicate the location of the input slot mask in the assembly. The preceding size field shall be required. An input slot mask is an array of bits which represents inclusion or exclusion of a modules target to originator data in this adapter rack connection. Bit 0 in this array represents slot 0, bit 1 represents slot 1, etc.

The "**InputSlotMask1**" keyword shall indicate the location of the input slot mask in the assembly. The preceding size field shall be required. An input slot mask is an array of bits which represents inclusion or exclusion of a modules target to originator data in this adapter rack connection. Bit 0 in this array represents slot 1, bit 1 represents slot 2, etc.

The "**OutputSlotMask0**" keyword shall indicate the location of the output slot mask in the assembly. The preceding size field shall be required. An output slot mask is an array of bits which represents inclusion or exclusion of a modules originator to target data in this adapter rack connection. Bit 0 in this array represents slot 0, bit 1 represents slot 1,etc.

The "**OutputSlotMask1**" keyword shall indicate the location of the output slot mask in the assembly. The preceding size field shall be required. An output slot mask is an array of bits which represents inclusion or exclusion of a modules originator to target data in this adapter rack connection. Bit 0 in this array represents slot 1, bit 1 represents slot 2 etc.

The "**ConfigSlotMask0**" keyword shall indicate the location of the configuration slot mask in the assembly. The preceding size field shall be required. A configuration slot mask is an array of bits which represents inclusion or exclusion of a modules configuration data in the fwd_open of this adapter rack connection. Bit 0 in this array represents slot 0, bit 1 represents slot 1, etc.

The "**ConfigSlotMask1**" keyword shall indicate the location of the configuration slot mask in the assembly. The preceding size field shall be required. A configuration slot mask is an array of bits which represents inclusion or exclusion of a modules configuration data in the fwd_open of this adapter rack connection. Bit 0 in this array represents slot 1, bit 1 represents slot 2, etc.

The "**ExternalID**" combined with a number (decimal) keyword intended purpose is to allow individual device keying for adapter rack connections. It shall indicate the usage of the device "**ExternalID**" value in the case when device keying is desired or the **"GenericID"** value in the case when module keying is not desired. The decimal number specifies the slot of this "**ExternalID**".


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The "**ProxyAssem", "ProxyAssemExa", "ProxiedAssem" and "ProxiedAssemExa"** keywords shall be used to describe assemblies that are proxied by a CIP adapter device to another device that does not support the CIP protocol. An example of this is a ControlNet adapter module (the device proxying the connection) in a multiple slot I/O rack with an analog I/O module (the device the connection is proxied for).

The "**ProxyAssem"/"ProxyAssemExa"** keywords shall exist in the EDS for the device that performs the proxy; the "**ProxiedAssem"/"ProxiedAssemExa"** keywords shall exist in the EDS for the device that the proxy is performed for. The information in the [Modular] section shall be used to associate EDS files containing "**ProxyAssem"/"ProxyAssemExa"** keywords to EDS files containing "**ProxiedAssem**"/"**ProxiedAssemExa**" keywords. This association shall exist when both EDS files specify a matching Rack entry. "ProxyAssem"/"ProxiedAssem"/"ProxyAssemExa"/ProxiedAssemExa" and "Assem"/"AssemExa" number ranges are independent of each other.

The decimal number (that is combined with "**ProxyAssem**"/"**ProxyAssemExa**" and "**ProxiedAssem**"/"**ProxiedAssemExa**") shall be used to match a "**ProxyAssem**"/"**ProxyAssemExa**" to a "**ProxiedAssem"**/"**ProxiedAssemExa**". The field values of a matched "**ProxyAssem"**/"**ProxyAssemExa**" and "**ProxiedAssem"**/"**ProxiedAssemExa**" pair shall be combined to constitute the same field value information that exists in a single "**Assem**"/"**AssemExa**" entry. This combination shall be done by using the field value from the "**ProxyAssem"**/"**ProxyAssemExa**" unless that field value is one of the keywords "**Module**" or "**ModuleMemberList**". When the field value specified in the "**ProxyAssem"**/"**ProxyAssemExa**" is "**Module**" the field value specified in the "**ProxiedAssem"**/"**ProxiedAssemExa**" shall be used. The field value "**Module"** shall not be used for "Member Size" or "Member Reference" fields. "**ModuleMemberList"** shall only be used in place of a "Member Size" and "Member Reference" field pair. When the field value specified in the "**ProxyAssem"**/"**ProxyAssemExa**" is "**ModuleMemberList"** all "Member Size" and "Member Reference" fields specified in the "**ProxiedAssem**"/"**ProxiedAssemExa**" shall be used. It shall be legal to specify field values for "**ProxiedAssem"**/"**ProxiedAssemExa**" entries whose corresponding field value in the "**ProxyAssem"**/"**ProxyAssemExa**" is not "**Module"**, however, these field values shall not by used, they shall exist only for documentation.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 7-3.32 Example of ProxyParam and ProxyAssem**

```
[Params]
   Param1 = 0,,,0x0010,0xC7,2,” Target Error Codes",
            "","",0,0xFFFF,0,0,0,0,0,0,0,0,0,0;
   ProxyParam1 = 0,,,0x0000,0xC7,2,"input size",
                  "","",Module,Module,Module,0,0,0,0,,,,,0;
   ProxyParamSizeAdder1 = 4,4,4;

[Assembly]
   Assem1 = "connection input format",,,,,,
            32,Param1,
            ,ProxyAssem1,
            ,ProxyAssem2;
   ProxyAssem1 = "real time input format","20 7D 24 SLOT 30 0A",,,,,
            ModuleMemberList;
   ProxyAssem2 = "real time status format","20 7D 24 SLOT 30 0B",,,,,
            ModuleMemberList;

[Connection Manager]
   ProxyConnect1 = 0x010100002, 0x44244401,
            2, 0, ,  2, ProxyParam1, Assem1, , , , , "Listen Only", "",
            "01 SLOT_MINUS_ONE 20 04 24 03 2C 04 2C 02";
```

**Figure 7-3.33 Example of ProxiedParam and ProxiedAssem**

```
[Params]
   ProxiedParam1  = ,,,,,,"input size","","",0,0xC7,2,,,,,,,,,;

[Assembly]
   ProxiedAssem1 = "real time input format",,,,,;
   ProxiedAssem2 = "real time status format",,,,,,16,;

[Connection Manager]
   ProxiedConnect1 = ,,,0,,,,,,,,,,,;
```

For bit fields specified in proxy keyword fields, the binary value shall allow a character of M (in addition to the 0 and 1 characters). The character M shall indicate to use the character specified in the proxied keyword field.

For example, the descriptor bits for a ProxyAssemN whose value is 0b0000000000000001M shall indicate that the assembly is a fixed sized expanded rack assembly and the ProxiedAssemN defines the allow value edit setting.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 7-3.7.2.5 Additions to the Connection Manager Section

The "**ProxyConnect"** and **"ProxiedConnect"** keywords shall be used to describe connections that are proxied by a CIP adapter device to another device that does not support the CIP protocol. An example of this is a ControlNet adapter module (the device proxying the connection) in a multiple slot I/O rack with an analog I/O module (the device the connection is proxied for).

The "**ProxyConnect"** keyword entry shall exist in the EDS for the device that performs the proxy. In the example above, this would be the ControlNet adapter module.

The "**ProxiedConnect"** keyword entry shall exist in the EDS for the device that the proxy is performed for. In the example above, this would be the analog I/O module.

The information in the [Modular] section shall be used to associate EDS files containing "**ProxyConnect"** keywords to EDS files containing "**ProxiedConnect**" keywords. This association shall exist when both EDS files specify a matching **Rack** entry. "ProxyConnect"/"ProxiedConnect" and "Connection" number ranges are independent of each other.

The decimal number (that is combined with "**ProxyConnect**" and "**ProxiedConnect**") shall be used to match a "**ProxyConnect**" to a "**ProxiedConnect"**. The field values of a matched "**ProxyConnect"** and "**ProxiedConnect"** pair shall be combined to constitute the same field value information that exists in a single "**Connection**" entry. This combination shall be done by using the field values from the "**ProxyConnect"** except for those fields where the value is the keyword "**Module"**. In those cases, the field value specified in the associated "**ProxiedConnect"** shall be used. It shall be legal to specify field values for "**ProxiedConnect"** entries whose corresponding field value in the "**ProxyConnect"** entry is not "**Module"**, however, these field values shall not be used, they shall exist only for documentation.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This page is intentionally left blank


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Volume 1: Common Industrial Protocol (CIP™) Specification

# Chapter 8: Physical Layer


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Contents

8-1 Introduction....3
8-2 Introduction to Auxiliary Power Distribution System....3
8-3 Transmission Media....3
8-3.1 Auxiliary Power Topology....3
8-3.2 Connectors....5
8-3.3 Device Taps....5
8-3.4 Power Taps....5
8-3.5 Grounding....6
8-3.6 Media....6
8-3.7 Connector Profiles....11
8-3.8 Auxiliary Tap Specifications....20
8-3.9 Power Tap Profiles....23
8-3.10 Device Auxiliary Power Port Specifications....25
8-3.11 Power Supply Specifications....26
8-3.12 Labeling Requirements....26
8-3.13 Vendor Declaration....26


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 8-1 Introduction

The CIP application layer can be used on a variety of network technologies. Each CIP network specification consists of two volumes. See the appropriate CIP network adaptation volume for physical layer behavior defined on that network.

Section 8-2 defines an optional auxiliary power distribution system that is separate and distinct from the physical layer requirements of any of the CIP networks.

# 8-2 Introduction to Auxiliary Power Distribution System

Auxiliary power may be used to provide application power for such devices as Input/Output modules, Emergency Stop circuitry, and other application specific needs. In some cases, it may provide additional power to communication devices at the application level. The cabling system provides 4 wire, two-circuit wiring that supplies 24V switched and un-switched power. Depending on the cabling selected by the designer, the maximum current ranges from 7 to 10 amperes. This standard specifies system topologies, cable and connector requirements and power supply requirements for auxiliary power distribution. The following clauses specify the allowable minimum and maximum allowable voltages within the system.

This system is not intended to provide redundant network power for already powered networks such as DeviceNet and or CompoNet.

The following table describes the function for each circuit within the auxiliary power bus and devices:

| Function of circuit | Circuit description |
|---|---|
| V+ Switched | +24 VDC power to circuits or devices, that may required a power disconnect (e.g. Emergency stop circuitry or the solenoids in pneumatic valve) while communications are kept active via the “Unswitched” circuit. |
| V- Switched Common | Connected to V- of the switched power supply commonly known as 0 VDC -24 VDC or common. |
| V+ Unswitched | +24 VDC unswitched power to circuits or devices that need power continuously (e.g. Sensor, or communication circuitry of pneumatic valve). |
| V- Unswitched Common | Connected to V- of the unswitched power supply commonly known as 0 VDC, -24 VDC, or common. |
| These four circuits shall not be connected to each other within the cabling components or internal in the node. | |

# 8-3 Transmission Media

## 8-3.1 Auxiliary Power Topology

The topology can be one or combination of the following topologies:

### 8-3.1.1 Trunk drop topology

Each device shall connect to the trunk through a tap.

### 8-3.1.2 Daisy chain topology

Each device shall provide connectivity for the trunk either by means of two connectors (one male and one female) or one connector with a direct connected tap. Power shall be applied to the male connector and shall provide power to the network through a female connector. When only one connector is present on a device, it shall be a male.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 8-3.1.3 Star topology

Power shall be provided to the devices through a Power distribution box consisting of a number of female jacks. Jumpers between the power distribution box and the device shall have male plugs on one end and female jacks on the device end.

**Figure 8-3.1 Trunk Drop Topology**

**Figure 8-3.2 Daisy Chain Topology**

**Figure 8-3.3 Star Topology**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 8-3.1.4 Maximum Cable Length

The total length of cable from the power supply is dependent upon the voltage drop from power supply to the device. The total voltage drop allowed between the power supply and any device is limited to 6V. This guarantees a minimum of 18V to all devices. The calculation of voltage drop shall take into consideration both the cable and Tap DC resistance (DCR)

## 8-3.2 Connectors

All connectors shall support four conductors, which accommodate a V+ switched conductor, V+ unswitched conductor, V- switched common conductor, and V- unswitched common conductor. See 8-3.10 for pin outs for all auxiliary power connectors.

The connector should be keyed such that an auxiliary power cable exits the instrument or device without interfering with any indicators, or anything else that may require access in the field. See Table 8-3.7 and Table 8-3.9 for details.

**Important:** All exposed male pins shall not be energized until mated to a female connector.

**Important:** All nodes attaching to auxiliary power cabling with a connector shall have male pins.

**Important:** Auxiliary power shall be supplied by the female contacts.

**Important:** Five circuit female circular (jacks) connectors are not allowed as they can potentially interconnect with DeviceNet connectors. Open style connectors provide special keying to prevent mating with a DeviceNet 5 pin open style connector.

## 8-3.3 Device Taps

Device taps provide points of attachment for devices onto the trunk line, only for the trunk drop and daisy chain topologies. Devices can be connected to the network either directly to the tap or with drop line. Taps also provide easy removal of a device without disrupting auxiliary power to downstream devices. See section 8-3.8 for more details

## 8-3.4 Power Taps

A power tap connects the power supply to the power trunk line for power trunk drop topology only. In addition to the requirements of section 8-3.9, power taps differ from device taps in that they may contain the following:

- A Schottky Diode which connects to the power supply V+ and allows for multiple supplies to be connected (this eliminates the need for custom power supplies).
- Two fuses or circuit breakers (on the drop line side) to limit the auxiliary power source from excess current, which could damage the cable and connectors. (not shown in Figure 8-3.4)
- A continuous connection for V- switched and unswitched common
- Current limiting on both circuits in each direction from the tap (shown in Figure 8-3.4)


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 8-3.4 Typical Power Tap Schematic**

## 8-3.5 Grounding

Isolated or single point grounding may be provided for each power supply.

## 8-3.6 Media

### 8-3.6.1 Auxiliary Power Trunk Cable Profile

**Table 8-3.1 Auxiliary Power Trunk Cable Profile**

| Physical Characteristics | Specification |
|---|---|
| Conductor size | #16 (1.3mm$^2$ ) Copper (minimum): 19 strands minimum (individually tinned) |
| Insulation diameter | To be specified by vendor |
| Colors | Per country standard |
| **Electrical Characteristics** | **Specification** |
| DCR - @ 20 deg C | 4.9 Ohms/1000 ft (13.17 Ohms/km) (maximum) |
| **Physical Characteristics** | **Specification** |
| Outside diameter | Per application |
| Roundness | 20% maximum of O.D. |
| Jacket Marking | Vendor name and part # and additional markings as required by certifying agencies |
| **Environmental Characteristics** | **Specification** |
| Agency Certifications | Per country standard |
| Flexure | Per application |
| Bend Radius | Per application |
| Operating Temperature | 0 to 70 degrees C minimum |
| Storage Temperature | -40 to +85 degrees C minimum |
| Pull Tension | Per application |
| Connector Compatibility | Mini, Open |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 8-3.5 Maximum Current versus Length for Trunk Cable (16 AWG)**

**Table 8-3.2 Maximum Current versus Length for Trunk Cable (16 AWG)**

| Cable Length Meters (Feet) | Maximum Current (I) | | Cable Length Meters (Feet) | Maximum Current (I) |
|---|---|---|---|---|
| 0 (0) | 10.00 | | 260 (853) | 0.57 |
| 20 (66) | 7.14 | | 280 (919) | 0.53 |
| 40 (131) | 3.66 | | 300 (984) | 0.50 |
| 60 (197) | 2.46 | | 320 (1050) | 0.47 |
| 80 (262) | 1.85 | | 340 (1115) | 0.44 |
| 100 (328) | 1.49 | | 360 (1181) | 0.42 |
| 120 (394) | 1.24 | | 400 (1312) | 0.37 |
| 140 (459) | 1.06 | | 420 (1378) | 0.36 |
| 160 (525) | 0.93 | | 440 (1444) | 0.34 |
| 180 (591) | 0.83 | | 460 (1509) | 0.33 |
| 200 (656) | 0.75 | | 480 (1575) | 0.31 |
| 220 (722) | 0.68 | | 500 (1640) | 0.30 |
| 240 (787) | 0.62 | | | |

Based on NFPA 79 at 75 Deg C temperature and a maximum of 10 Amps. In addition the power bus is limited by a recommended minimum 18 Volts delivered to load.

**Important:** Some devices may not work with an operating voltage as low as 18 Volts. Designer must consider this during system layout for auxiliary powered devices.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 8-3.6.2 Auxiliary Power Drop Cable Profile

**Table 8-3.3 Auxiliary Power Drop Cable Profile**

| Physical Characteristics | Specification |
|---|---|
| Conductor size | #18 Copper (minimum): 19 strands minimum (individually tinned) |
| Insulation diameter | To be specified by vendor |
| Colors | Per country standard |
| **Electrical Characteristics** | **Specification** |
| DCR - @ 20 deg C | 6.9 Ohms/1000 ft (maximum) |
| **Physical Characteristics** | **Specification** |
| Outside diameter | Per application |
| Roundness | 20% maximum of O.D. |
| Jacket Marking | Vendor name and part # and additional markings as required by certifying agencies |
| **Environmental Characteristics** | **Specification** |
| Agency Certifications (U.S. and Canada) | Per country standard |
| Flexure | Per application |
| Bend Radius | Per application |
| Operating Temperature | 0 to 70 degrees C minimum |
| Storage Temperature | -40 to +85 degrees C minimum |
| Pull tension | Per application |
| Connector Compatibility | Mini, Micro |

**Figure 8-3.6 Maximum Current versus Length for Drop Cable**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 8-3.4 Maximum Current versus Length for Drop Cable**

| Cable Length Meters (Feet) | Maximum Current (I) | | Cable Length Meters (Feet) | Maximum Current (I) |
|---|---|---|---|---|
| 0 (0) | 7.00 | | 260 (853) | 0.36 |
| 20 (66) | 4.57 | | 280 (919) | 0.34 |
| 40 (131) | 2.32 | | 300 (984) | 0.31 |
| 60 (197) | 1.56 | | 320 (1050) | 0.29 |
| 80 (262) | 1.17 | | 340 (1115) | 0.28 |
| 100 (328) | 0.94 | | 360 (1181) | 0.26 |
| 120 (394) | 0.78 | | 400 (1312) | 0.24 |
| 140 (459) | 0.67 | | 420 (1378) | 0.22 |
| 160 (525) | 0.59 | | 440 (1444) | 0.21 |
| 180 (591) | 0.52 | | 460 (1509) | 0.20 |
| 200 (656) | 0.47 | | 480 (1575) | 0.20 |
| 220 (722) | 0.43 | | 500 (1640) | 0.19 |
| 240 (787) | 0.39 | | | |

Based on NFPA 79 at 75 Deg C temperature and a maximum of 7 Amps. In addition the power bus is limited by a recommended minimum 18 Volts delivered to load.

**Important:** Some devices may not work with an operating voltage as low as 18 Volts. Designer must consider this during system layout for auxiliary powered devices.

### 8-3.6.3 Auxiliary Power Flat Trunk Cable Profile

**Table 8-3.5 Auxiliary Power Flat Trunk Cable Profile**

| Physical Characteristics | Specification |
|---|---|
| Conductor size | #16 Copper (minimum): 19 strands minimum (individually tinned) |
| Insulation diameter | To be specified by vendor |
| Colors | Per country standard |
| **Electrical Characteristics** | **Specification** |
| DCR - @ 20 deg C | 4.9 Ohms/1000 ft (maximum) |
| **Physical Characteristics** | **Specification** |
| Outside diameter | Per country standard |
| Roundness | N/A |
| Jacket Marking | Vendor name and part # and additional markings as required by certifying agencies |
| **Environmental Characteristics** | **Specification** |
| Agency Certifications (U.S. and Canada) | Per application |
| Flexure | Per application |
| Bend Radius | Per application |
| Operating Temperature | 0 to 70 degrees C minimum |
| Storage Temperature | -40 to +85 degrees C minimum |
| Pull tension | Per application |
| Connector Compatibility | Flat |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 8-3.7 Maximum Current versus Length for Flat Trunk Cable**

**Table 8-3.6 Maximum Current versus Length for Flat Trunk Cable**

| Cable Length Meters (Feet) | Maximum Current (I) | Cable Length Meters (Feet) | Maximum Current (I) |
|---|---|---|---|
| 0 (0) | 10.00 | 260 (853) | 0.57 |
| 20 (66) | 7.14 | 280 (919) | 0.53 |
| 40 (131) | 3.66 | 300 (984) | 0.50 |
| 60 (197) | 2.46 | 320 (1050) | 0.47 |
| 80 (262) | 1.85 | 340 (1115) | 0.44 |
| 100 (328) | 1.49 | 360 (1181) | 0.42 |
| 120 (394) | 1.24 | 400 (1312) | 0.37 |
| 140 (459) | 1.06 | 420 (1378) | 0.36 |
| 160 (525) | 0.93 | 440 (1444) | 0.34 |
| 180 (591) | 0.83 | 460 (1509) | 0.33 |
| 200 (656) | 0.75 | 480 (1575) | 0.31 |
| 220 (722) | 0.68 | 500 (1640) | 0.30 |
| 240 (787) | 0.62 | | |

Based on NFPA 79 at 75 Deg C temperature and a maximum of 10 Amps. In addition the power bus is limited by a recommended minimum 18 Volts delivered to load.

**Important:** Some devices may not work with an operating voltage as low as 18 Volts. Designer must consider this during system layout for auxiliary powered devices.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 8-3.7 Connector Profiles

**Important:** Connectors shall not be connected or disconnected under load.

### 8-3.7.1 Sealed 7/8” (Mini) Connector

**Table 8-3.7 Sealed7/8” (Mini) Connector**

| Male General Characteristics | Specification |
|---|---|
| Number of pins (circuits1) | 4 |
| Coupling Nut | Male |
| Coupling Nut Thread | 7/8-16 UN-2A THD |
| Rotation | Optional |
| Standard | See Figure 8-3.10 for geometry |
| Pinout (Conductor order) | V+Switched: Pin1, V+ Unswitched: Pin 2, V- Unswitched Common: Pin 3, V- Switched Common: Pin 4 |
| **Female General Characteristics** | **Specification** |
| Number of sockets (circuits1) | 4 |
| Coupling Nut | Female |
| Coupling Nut Thread | 7/8-16 UN-2B THD |
| Rotation | Required |
| Standard | See Figure 8-3.10 for geometry |
| Pinout (Conductor Order) | V+Switched: Pin1, V+ Unswitched: Pin 2,V- Unswitched Common: Pin 3, V- Switched Common: Pin 4 |
| **Physical Characteristics** | **Specification** |
| Wiping Contact Plating Requirements | 30 micro inch (0.76µm) gold minimum over 50 micro inch (1.27µm) nickel minimum or 5 micro inch (0.127µm) gold minimum over 20 micro inch (0.508µm) palladium-nickel minimum over 50 micro inch (1.27µm) nickel. All gold shall be 24 Karat. |
| Wiping Contact Life | 1000 insertion-extractions (Not under load) |
| **Electrical Characteristics** | **Specification** |
| Operating Voltage | 30 volt minimum |
| Contact Rating | 8 Amps Min. |
| Contact Resistance | Nominal: Less than 1 mOhm<br>Maximum: 5 mOhm over life |
| **Environmental Characteristics** | **Specification** |
| Water Resistance | IP67 and NEMA 4, 6, 6P, 13 |
| Oil Resistance | UL-1277, OIL RES II |
| Operating ambient temperature | -40 to +70 deg C full power with linear derating to 0 amps at 80 deg C. |
| Storage temperature | -40 to +85 deg C |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 8-3.8 Mini Connector Pinout**

**Figure 8-3.9 Mini 90 Degree Connector**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 8-3.10 Recommended Dimensions for Four Pole Interface**

All dimensions are in inches


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 8-3.7.2 Sealed M12 (Micro) Connector

**Table 8-3.8 Sealed M12 (Micro) Connector**

| Male General Characteristics | Specification |
|---|---|
| Number of pins (circuits1) | 4 |
| Coupling Nut | Male |
| Coupling Nut Thread | M12 x 1 6G fit |
| Rotation | Optional |
| Standard | IEC 61076-2-101 |
| Pinout (Conductor order) | V+Unswitched: Pin1, V+ Switched: Pin 2, V- Unswitched Common: Pin 3, V- Switched Common: Pin 4 |
| **Female General Characteristics** | **Specification** |
| Number of sockets (circuits1) | 4 |
| Coupling Nut | Female |
| Coupling Nut Thread | M12 x 1 6G fit |
| Rotation | Required |
| Standard | IEC 61076-2-101 |
| Pinout (Conductor Order) | V+Unswitched: Pin1, V+ Switched: Pin 2, V- Unswitched Common: Pin 3,V- Switched Common: Pin 4 |
| **Physical Characteristics** | **Specification** |
| Wiping Contact Plating Requirements | 30 micro inch (0.76µm) gold minimum over 50 micro inch (1.27µm) nickel minimum or 5 micro inch (0.127µm) gold minimum over 20 micro inch (0.508µm) palladium-nickel minimum over 50 micro inch (1.27µm) nickel. All gold shall be 24 Karat. |
| Wiping Contact Life | 1000 insertion-extractions (Not under load) |
| **Electrical Characteristics** | **Specification** |
| Operating Voltage | 30 volt minimum |
| Contact Rating | 4 Amps Min. |
| Contact Resistance | Nominal: Less than 1 mOhm<br>Maximum: 5 mOhm over life |
| **Environmental Characteristics** | **Specification** |
| Water Resistance | IP67 and NEMA 4, 6, 6P, 13 |
| Oil Resistance | UL-1277, OIL RES II |
| Operating ambient temperature | -40 to +70 deg C full power with linear derating to 0 amps at 80 deg C. |
| Storage temperature | -40 to +85 deg C |

**Figure 8-3.11 Micro Connector Pinout**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 8-3.12 Micro 90 Degree Connector View**

Male 90 degree view

Female 90 degree view

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 8-3.7.3 Open Style Connector

**Table 8-3.9 Open Style Connector**

| Male General Characteristics | Specification |
|---|---|
| Number of pins (circuits1) | 5 |
| Coupling Nut | None |
| Coupling Nut Thread | None |
| Rotation | None |
| Standard | See Figure 8-3.13 for geometry |
| Pinout (Conductor order) | V+Unswitched: Pin1, V+ Switched: Pin 2, V- Unswitched Common: Pin 3, V- Switched Common: Pin 4, Not Used: Pin 5 |
| **Female General Characteristics** | **Specification** |
| Number of sockets (circuits1) | 5 |
| Coupling Nut | None |
| Coupling Nut Thread | None |
| Rotation | None |
| Standard | See Figure 8-3.13 for geometry |
| Pinout (Conductor Order) | V+Unswitched: Pin1, V+ Switched: Pin 2, V- Unswitched Common: Pin 3, V- Switched Common: Pin 4, Not Used: Pin 5 |
| **Physical Characteristics** | **Specification** |
| Wiping Contact Plating Requirements | 30 micro inch (0.76µm) gold minimum over 50 micro inch (1.27µm) nickel minimum or 5 micro inch (0.127µm) gold minimum over 20 micro inch (0.508µm) palladium-nickel minimum over 50 micro inch (1.27µm) nickel. All gold shall be 24 Karat. |
| Wiping Contact Life | 100 insertion-extractions (Not under load) |
| **Electrical Characteristics** | **Specification** |
| Operating Voltage | 30 volt minimum |
| Contact Rating | 8 Amps Min. |
| Contact Resistance | Nominal: Less than 1 mOhm<br>Maximum: 5 mOhm over life |
| **Environmental Characteristics** | **Specification** |
| Water Resistance | None |
| Oil Resistance | None |
| Operating ambient temperature | -40 to +70 deg C full power with linear derating to 0 amps at 80 deg C. |
| Storage temperature | -40 to +85 deg C |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 8-3.13 Open Style Connector**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 8-3.7.4 Flat Connector

**Table 8-3.10 Flat Connector**

| Male General Characteristics | Specification |
|---|---|
| Number of pins (circuits1) | 4 |
| Coupling Nut | (Optional retention with #8 x 1¾ inch screw) |
| Coupling Nut Thread | (UNC 32) |
| Rotation | None |
| Standard | See Figure 8-3.14 |
| Pinout (Conductor order) | |
| **Female General Characteristics** | **Specification** |
| Number of sockets (circuits1) | 4 |
| Coupling Nut | (Optional retention with #8 insert or nut) |
| Coupling Nut Thread | (UNC 32) |
| Rotation | None |
| Standard | See Figure 8-3.14 |
| Pinout (Conductor Order) | |
| **Physical Characteristics** | **Specification** |
| Wiping Contact Plating Requirements | 8 micro inch (0.2µm) gold minimum over 50 micro inch (1.27µm) nickel minimum. All gold shall be 24 Karat. |
| Wiping Contact Life | 100 insertion-extractions |
| **Electrical Characteristics** | **Specification** |
| Operating Voltage | 25 volt minimum |
| Contact Rating | 3 Amps Min. |
| Contact Resistance | Nominal: Less than 1 mOhm<br>Maximum: 5 mOhm over life |
| **Environmental Characteristics** | **Specification** |
| Water Resistance | IP67 and NEMA 4, 6, 6P, 13 |
| Oil Resistance | UL-1277, OIL RES II |
| Operating ambient temperature | -40 to +70 deg C full power with linear derating to 0 amps at 80 deg C. |
| Storage temperature | -40 to +85 deg C |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 8-3.14 Flat Connector Pin-out**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 8-3.8 Auxiliary Tap Specifications

### 8-3.8.1 Mini Trunk to Mini Drop Tap profile

**Table 8-3.11 Mini Trunk to Mini Drop Tap Profile**

| Internal Pass Through Conductor Description | Specification |
|---|---|
| Length of conductors | 4 inches (101.6mm) (maximum) |
| Conductor resistance@ 20ºC | 1.2 mOhms (maximum) |
| Conductor equivalent | 18 AWG (0.823mm$^2$) (minimum) |
| **Internal Drop Conductor Description** | **Specification** |
| Length of conductors | 7 inches (177.8mm) (maximum) |
| Conductor resistance @ 20ºC | 4 mOhms (maximum) |
| Conductor equivalent | 18 AWG (0.823mm$^2$) (minimum) |
| **Electrical Characteristics** | **Specification** |
| Operating Voltage | 25 Volt (minimum) |
| Trunk Contact Rating | 8 amps (minimum) |
| Drop Contact Rating | 8 amps (minimum) |
| Contact DCR | See section 8-3.7 |
| **Environmental Characteristics** | **Specification** |
| Water resistance | IP67 and NEMA 4, 6, 6P, 13 |
| Oil resistance | UL 1277, OIL RES II |
| Operating ambient temperature | -40 to +70 C @ 8 amps; de-rate current linearly to 0 @ 80C |
| Storage temperature | -40 to +85C |
| Marking/Labeling | Vendor name and additional marking |
| The total allowable DCR from port to port is equal to the contact DCR + internal cable DCR. | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 8-3.15 Mini Trunk to Mini Drop Tap Keying Orientation**

## 8-3.8.2 Mini Trunk to Micro Drop Tap profile

**Table 8-3.12 Mini Trunk to Micro Drop Tap Profile**

| Internal Pass Through Conductor Description | Specification |
|---|---|
| Length of conductors | 4 inches (101.6mm) (maximum) |
| Conductor resistance | 1.2 mOhms (maximum) |
| Conductor equivalent | 18 AWG (0.823mm$^2$ ) (minimum) |
| **Internal Drop Conductor Description** | **Specification** |
| Length of conductors | 7 inches (177.8mm) (maximum) |
| Conductor resistance | 10 mOhms (maximum) |
| Conductor equivalent | 22 AWG (0.326mm$^2$) (minimum) |
| **Electrical Characteristics** | **Specification** |
| Operating Voltage | 25 Volt (minimum) |
| Trunk Contact Rating | 8 amps (minimum) |
| Drop Contact Rating | 4 amps (minimum) |
| Contact DCR | See section 8-3.7 |
| **Environmental Characteristics** | **Specification** |
| Water resistance | IP67 and NEMA 4, 6, 6P, 13 |
| Oil resistance | UL 1277, OIL RES II |
| Operating ambient temperature | -40 to +70 C @ 8 amps; de-rate current linearly to 0 @ 80C |
| Storage temperature | -40 to +85C |
| Marking/Labeling | Vendor name and additional marking |
| Recommend Keying | See figure below |
| The total allowable DCR from port to port is equal to the contact DCR + internal cable DCR. | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 8-3.16 Mini Trunk to Micro Drop Tap Keying Orientation**

## 8-3.8.3 Micro Trunk to Micro Drop Tap profile

**Table 8-3.13 Micro Trunk to Micro Drop Tap Profile**

| Internal Pass Through Conductor Description | Specification |
|---|---|
| Length of conductors | 4 inches (101.6mm) (maximum) |
| Conductor resistance | 1.2 mOhms (maximum) |
| Conductor equivalent | 18 AWG (0.823mm$^2$) (minimum) |
| **Internal Drop Conductor Description** | **Specification** |
| Length of conductors | 7 inches (177.8mm) (maximum) |
| Conductor resistance | 10 mOhms (maximum) |
| Conductor equivalent | 22 AWG (0.326mm$^2$) (minimum) |
| **Electrical Characteristics** | **Specification** |
| Operating Voltage | 25 Volt (minimum) |
| Trunk Contact Rating | 4 amps (minimum) |
| Drop Contact Rating | 4 amps (minimum) |
| Contact DCR | See section 8-3.7 |
| **Environmental Characteristics** | **Specification** |
| Water resistance | IP67 and NEMA 4, 6, 6P, 13 |
| Oil resistance | UL 1277, OIL RES II |
| Operating ambient temperature | -40 to +70 C @ 8 amps; de-rate current linearly to 0 @ 80C |
| Storage temperature | -40 to +85C |
| Marking/Labeling | Vendor name and additional marking |
| Recommend Keying | See figure below |
| The total allowable DCR from port to port is equal to the contact DCR + internal cable DCR. | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure 8-3.17 Micro Trunk to Micro Drop Tap Keying Orientation**

## 8-3.9 Power Tap Profiles

### 8-3.9.1 Power Tap Specifications

Included are the following tables regarding Auxiliary Power Taps:

- General Power Tap Specifications
- Internal Pass Through Conductor Specifications
- Internal Power Drop Conductor Specifications
- Electrical Specifications
- Environmental Specifications

**Table 8-3.14 Power Tap Specifications**

| Physical Characteristic | Specification |
|---|---|
| Marking/Labeling | Power Tap, Vendor name & part # |
| Trunk connectors | Sealed or open per section 8-3.10 |
| Power Drop connectors | Sealed or open per section 8-3.10 |
| Protection circuit (times 2) (optional) | Non-auto reset fuse or circuit breaker in power tap |
| Protection serviceability (optional) | Field resettable/replaceable |
| Shield ground connection | 10-32 screw or stud on case or grounding wire |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### 8-3.9.2 Connector Gender Keying

**Figure 8-3.18 Gender and Keying of Power Tap**

Note: The defined connector gender prevents any exposed powered male contacts with single power supply auxiliary power distribution systems. Certain applications might require different connector gender options than shown above, to minimize exposed power male pins.

When adding or relocating power supplies on an existing auxiliary power distribution system, between male-female connections cordsets, one of the trunk connectors may be a male connector as specified in section 8-3.2.

### 8-3.9.3 Typical Power Tap Schematic

**Figure 8-3.19 Typical Power Tap Schematic**

**Table 8-3.15 Internal Pass Through Conductor Specifications:**

| Conductor Description | Specifications |
|---|---|
| Power Conductors | 7 inches (177.8mm) maximum conductor length. Maximum conductor resistance of 2.1 mOhms (Equivalent to 7” (177.8mm) of 15 AWG (1.65 $mm^2$). Minimum conductor equivalent 16 AWG (1.30$mm^2$) |

**Table 8-3.16 Internal Power Drop Conductor Specifications:**

| Conductor Description | Specifications |
|---|---|
| Power Conductors | 12 inches (304.8mm) maximum conductor length per leg. Maximum resistance per leg (excluding protection) 3.6 mOhms (Equivalent to 12” (304.8mm) of 15 AWG (1.65 $mm^2$)). Minimum conductor equivalent at any location 16 AWG (1.30$mm^2$). Any conductor carrying between 8 and 16 amps shall be 12 AWG (303 $mm^2$) equivalent or larger. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 8-3.17 Electrical Specifications**

| Electrical Characteristics | Specification |
|---|---|
| Protection Circuit (Optional) | Hold, 8 amps for trunk, 4 amps for drop<br>1.5 sec min. to 120 sec max. trip time for 20 amps at 25 deg C.<br>Slo-Blo or Normal-Blo Allowable |
| Schottky Diode (Optional) | 20 amp, 30 volt minimum continuous capability over environment. Max Vf of .73 volts @ Tc=125 C and 16 A minimum test condition.<br>MBR3045 Recommended |
| Operating Voltage | 25 Volt minimum |
| Contact Rating | Trunk: 8 amps min.<br>Drop: 3 amps min.<br>Over temperature for all contacts.<br>Drop line: The sum of the contact ratings for both V+ and V- shall be 16 amps minimum for thick drop and 6 amps minimum for thin drop over temperature. |
| Contact DCR | See Section 8-3.7 |

**Table 8-3.18 Environmental Specifications**

| Electrical Characteristics | Specification |
|---|---|
| Water resistance (Sealed) | IP67 and NEMA 4, 6, 6P, 13 for sealed applications |
| Oil resistance (Sealed) | UL-1277, OIL RES II for sealed applications |
| Operating ambient temperature | -40 to +70 C with maximum continuous power on all conductors. De-rate linearly to 0 amps at 80 deg C |
| Storage temperature | -40 to +85 C |

Note: The use of manufacturer's names and catalog numbers may occur in these specifications. These occurrences do not imply or constitute qualification. These occurrences are intended as a temporary measure and will exist only until replacement standards are developed.

### 8-3.9.4 Multiple Port Power Taps

Multiple port power taps are permissible. A power tap with multiple ports shall contain one male connector with 2 or more female connectors. The connectors (male and female) shall be labeled as specified in Table 8-3.19.

## 8-3.10 Device Auxiliary Power Port Specifications

The following requirements shall be met by devices claiming conformance to this specification:

### 8-3.10.1 Connectors

Device shall contain a minimum of one connector. When one connector is supported, that connector shall be a male connector. When two connectors are supported by the device, the second connector shall be a female.

### 8-3.10.2 Allowable DCR for Devices with Two ports

When a device supports two ports, The DCR from one port to another shall be less than or equal to 3.6 milliohms + the applicable connector DCR as described in Section 8-3.7.

### 8-3.10.3 Device Labeling

The auxiliary ports shall be labeled in accordance with Table 8-3.19 below.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 8-3.19 Auxiliary Power Port Labeling**

| Connector | Label |
|---|---|
| Female | 24V Out, optional 24V AUX PWR Out |
| Male | 24V In, optional 24V AUX PWR In |

Labels shall be permanent.

## 8-3.11 Power Supply Specifications

The following table outlines the minimum requirements of a power supply that is to be connected to Auxiliary power network. These specifications apply to single supply and multiple supplies.

**Table 8-3.20 Auxiliary Power Supply Specifications**

| Specification | Parameter |
|---|---|
| Initial Tolerance | 24 volts ± 1% for fixed output power supplies<br>Adjustable output power supplies with output range of 22-27.2 VDC |
| Line Regulation | 0.5% max |
| Load Regulation | 1.5% max |
| Temperature Coefficient | 0.05% per deg C max from 20 deg C to 50 deg C |
| Output Ripple | 480 mV p-p |
| Temperature Range | Operating 0 to 60 deg C<br>Storage -25 to 65 deg C<br>*De-rating acceptable for 40 deg C operation |
| Inrush Current Limit | Less than 65 Amps peak |
| Output Over Voltage Protection | 36 Vdc maximum |
| Output Over Current Protection | Yes (current limit 160% max) |
| Turn-on Time (with full load) | 1000 msec maximum. within 5% of final value |
| Turn-on Overshoot | 3.5% maximum |
| Stability | 0 to 100% load (all conditions) |
| Isolation | Output isolated from AC and Chassis ground |
| Output Voltage | 24 Volts ±1% |
| Output Current | Per application and per local regulation codes |
| Output Surge current capability | 10% reserve capability |

When the same power supply is used for both CIP network power and auxiliary power, then the network power supply requirements of the specific CIP network shall apply.

## 8-3.12 Labeling Requirements

Minimum labeling shall require marking of V+ and V- Switched positions on connectors and power supplies.

## 8-3.13 Vendor Declaration

The vendor shall declare in the datasheet/user manual if the device conforms to CIP Auxiliary power.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Volume 1: Common Industrial Protocol (CIP™) Specification

# Chapter 9: Indicators and Middle Layers


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Contents

9-1 Introduction ..... 3


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 9-1 Introduction

The CIP application layer can be used on a variety of network technologies. Each CIP network specification consists of two volumes. This volume is not currently specifying any common indicator or middle layer behavior. See the appropriate CIP network adaptation specifications for indicator or middle layer behavior defined on that network.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This page is intentionally left blank


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Volume 1: Common Industrial Protocol (CIP™) Specification

# Chapter 10: Bridging and Routing


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Contents

10-1 Introduction ..... 3
10-2 CIP Routing ..... 3
10-3 Object Addressing Space ..... 3
10-4 CIP Routing Behavior ..... 4
10-4.1 Connection Manager Events and Actions ..... 4
10-4.2 Routing to Non-CIP Networks ..... 5
10-5 DeviceNet Modifications ..... 5
10-5.1 Unconnected Send Service Request Modification ..... 6
10-5.2 Unconnected Send Service Response Modification ..... 7
10-6 Examples ..... 8
10-6.1 Using the Unconnected Send Service for Explicit Messaging – Single Hop ..... 8
10-6.2 Using the Unconnected Send Service For Explicit Messaging – Multiple Hop ..... 11
10-6.3 Using the Forward Open Service to Open an I/O Messaging Connection ..... 14
10-6.4 Using the Forward Close Service to Close an I/O Messaging Connection ..... 17


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 10-1 Introduction

The communication objects within Chapter 3, in particular services of the Connection Manager, provide bridging and routing between CIP subnets. This chapter will describe how bridging and routing is accomplished using those objects.

## 10-2 CIP Routing

The Forward_Open, Forward_Close, and Unconnected_Send services of the Connection Manager object allow for routing of messages and for the establishment of connections across bridges. Use of the Port Segment within the Connection Path parameter of the above services provides the routing information. The Port Segment encoding is described in Appendix C. Routing is 'fixed path' (data will always follow the same path); this is critical for insuring consistent I/O transaction times.

Each Port Segment represents a 'hop' in the path, from one subnet to another. The router processes and removes all segments (Electronic Key Segment, Network Segments) up to and including the first Port Segment from the path before forwarding the service to the next destination. The Port Segment tells the router which port (subnet) to send the message on and the node address of the destination node on that subnet. The final target receives a Connection Path with no Port Segments remaining, however, it may include other segments (Electronic Key Segment, Network Segments, etc.).

A connection originator can use the Port Object within each router to determine the types of subnets, which can be reached (routed to) through that device. The Port Object, described in Chapter 3, provides the port name, port number within the device, and other port information for each routable port on the device.

## 10-3 Object Addressing Space

Prior versions of the specification provided for port addressing space and device addressing space for object instances. The port addressing space definition is no longer supported; all object instances are in the device space.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 10-4 CIP Routing Behavior

This section describes various events and conditions that routers and gateways may encounter and the behaviors that shall be exhibited in those cases.

## 10-4.1 Connection Manager Events and Actions

The table below presents the Events that a Connection Manager Object instance may experience and the appropriate actions to take.

**Table 10-4.1 CIP Routing Event/Action Matrix**

| Event Description | Action to Take |
|---|---|
| Connection Manager Object Instance receives a valid Unconnected Send Request that DOES NOT need to be forwarded on through more intermediate hops. | Process and update the timing parameters[1]. Internally route the request to the specified port's protocol engine. Establish an Explicit Messaging Connection with the Remote Target Device. Extract the Explicit Messaging Request parameters from the Message Request portion of the Unconnected Send Service Data. Deliver the Explicit Messaging Request to the Remote Target Device. Start a "wait for response" timer whose value is the number of milliseconds specified by the current (updated) timing parameters. |
| Connection Manager Object Instance receives an invalidly formatted Unconnected Send Request. | Return a Routing Error that conveys the detected problem. See the Packet Validation section below for more details. |
| Failure to execute the Explicit Messaging transaction with the Remote Target Device. This is due to:<br>• Failure to establish an Explicit Messaging Connection<br>• Open Explicit Messaging Connection Error Response<br>• Timeout waiting for the Explicit Messaging Response. | Return a Routing Error with the following settings:<br>• General Status field is set to 01<br>• Additional Status array contains a single UINT value of 0x0204<br>• remainingPathSize is set to the value zero (0). |
| Failure to execute the Unconnected Send transaction with the next router in the path. This is due to:<br>• Failure to establish an Explicit Messaging Connection<br>• Open Explicit Messaging Connection Error Response<br>• Timeout waiting for the Unconnected Send Explicit Messaging Response. | Return a Routing Error with the following settings:<br>• General Status field is set to 01<br>• Additional Status array contains a single UINT value of 0x0204<br>• Remaining Path Size is set to the number of words remaining in the "pre-stripped" path. |
| Unconnected Send Request is received but it cannot be processed due to an internal resource error (e.g. queue overflow, no buffers available, etc.). | Return a Routing Error with the following characteristics:<br>• General Status field is set to 02<br>• Additional Status array is empty<br>• Remaining Path Size is set to the "pre-stripped" number of words remaining in the Route_Path field. |
| Connection Manager Object Instance receives a valid Unconnected Send Request that needs to be routed through more intermediate hops before it reaches its final destination network. | Process and update the timing parameters[1]. Internally route the request to the specified port's protocol engine. Strip off the appropriate portion of the routing information and update the Transaction_Id field if necessary. Establish an Explicit Messaging Connection with the next router and deliver the updated Unconnected Send Request accordingly. Start a "wait for response" timer whose value is the number of milliseconds specified by the current timing parameters (AFTER they have been updated). |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Event Description | Action to Take |
|---|---|
| Explicit Messaging Response OR an Unconnected Send Response is received by a router. | Determine whether or not the Explicit Messaging Connection across which the associated Unconnected Send Request is still active (did not experience an Inactivity/Watchdog timeout). Determine whether or not a timeout error has already been returned for this transaction. If the Explicit Messaging Connection is still active and a routing error has yet to be returned for this transaction, then formulate the appropriate Unconnected Send Response and return it across that Explicit Messaging Connection. This may entail restoring the requester's Transaction_Id field on DeviceNet. If a Routing Error has already been returned OR the Explicit Messaging Connection is no longer active, then the router shall discontinue all processing associated with this transaction. |
| Router experiences a timeout while waiting for an Explicit Messaging Response from either another router (to which an Unconnected Send request had been forwarded) or the Remote Target Device (to which the actual Explicit Messaging Request had been delivered). | Return a Routing Error with the following settings:<br>• General Status field is set to 01<br>• Additional Status array contains a single UINT value of 0x0204<br>• remainingPathSize is set to the number of words remaining in the "pre-stripped" path. |

Table Footnotes
1. Any time a Connection Manager Object receives an Unconnected Send Request it SHALL process the timing parameters.
2. Whenever an Unconnected Send Request/Response is transmitted/received, a counter within the Connection Manager Object Instance associated with the port across which this transmission/reception took place shall be incremented.

### 10-4.2 Routing to Non-CIP Networks

A CIP gateway may provide identity information for nodes on the downstream subnet. Often, non-CIP devices do not contain a serial number, do not report it on the subnet or support a serial number that is not compliant for CIP usage. If a CIP client requests, through a gateway, the Identity Object Serial Number attribute from a non-CIP target device that does not contain a serial number, the gateway shall report ('proxy') a value of zero.

## 10-5 DeviceNet Modifications

The services of the Connection Manager are not supported by DeviceNet nodes when it is the target of the request. Thus, only nodes which provide routing to/from DeviceNet support the Connection Manager object. When the target node of a message is on DeviceNet, these routers are required to translate the message into normal DeviceNet explicit messaging format. As a result, DeviceNet nodes require no changes to accept routed messages from other CIP subnets.

In order to allow these routers the ability to handle multiple outstanding transactions on DeviceNet, the Unconnected Send service of the Connection Manager is modified when traversing a DeviceNet subnet. The details are provided in sections 10-5.1 and 10-5.2 below.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 10-5.1 Unconnected Send Service Request Modification

When sent on a DeviceNet subnet, the Unconnected Send service data is prepended with a 16-bit transaction ID. This transaction identifier is generated by the requesting device and returned by the router along with the response from the target. The modified service request is as shown below.

**Table 10-5.1 Unconnected Send Service Parameters**

| Parameter Name | | Data Type | Description |
|---|---|---|---|
| Transaction_ID | | UINT | Used for transaction management in the DeviceNet Requesting Device and DeviceNet Routers. This field is used for transaction matching by message originators on DeviceNet not passed on to other subnets |
| Priority/Time_tick | | BYTE | Used to calculate request timeout information. |
| Time-out_ticks | | USINT | Used to calculate request timeout information. |
| Message_Request_Size | | UINT | Specifies the number of bytes in the Message Request. |
| Message_Request[1] | | Struct of | |
| | Service | USINT | Service code of the request. |
| | Request_Path_Size | USINT | The number of 16 bit words in the Request_Path field (next element). |
| | Request_Path | Padded EPATH | This is an array of bytes whose contents convey the path of the request (Class ID, Instance ID, etc.) for this transaction. |
| | Request_Data | Array of octet | Service specific data to be delivered in the Explicit Messaging Request. If no additional data is to be sent with the Explicit Messaging Request, then this array will be empty. |
| Pad | | USINT | Only present if Message_Request_Size is an odd value. |
| Route_Path_Size | | USINT | The number of 16 bit words in the Route_Path field. |
| Reserved | | USINT | Reserved byte. Shall be set to zero (0). |
| Route_Path | | Padded EPATH | Indicates the route to the Remote Target Device. |

Table Footnotes
1. This is the Message Router Request Format as defined in Chapter 2.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 10-5.2 Unconnected Send Service Response Modification

A DeviceNet router shall always return a successful response to an Unconnected Send service request. This success response may actually indicate that an error was encountered. This is necessary because additional error information is needed to handle routing errors and this additional information does not conform to the Error Response format defined for DeviceNet. As specified in Chapter 3, the Service code returned may be either the service code sent inside the Unconnected Send service data *or* the Unconnected Send service code. The 0x94 Error Response Service is never returned.

The modified successful and unsuccessful service responses are as shown below. Note that this is the data placed within the DeviceNet Service Data area; the reply service code is earlier in the packet.

**Table 10-5.2 Successful Unconnected Send Response**

| Parameter Name | Data Type | Description |
|---|---|---|
| Transaction_ID | UINT | Echoes the value received in the associated Unconnected Send Request. |
| General Status | USINT | This value is zero (0) for successful transactions. |
| Reserved | USINT | Shall be zero. |
| Service Response Data | Array of byte | This field contains the Explicit Messaging Service Data returned by the Target Device/Object. For example, this would contain Attribute Data in response to a Get_Attribute_Single request. If the Explicit Messaging response returned by the Target Device/Object did not contain any Service Data, then this field shall be empty. |

The response Service Data associated with an unsuccessful Unconnected Send response is defined below.

**Table 10-5.3 Unsuccessful Unconnected Send Response**

| Parameter Name | Data Type | Description |
|---|---|---|
| Transaction_ID | UINT | Echoes the value received in the associated Unconnected Send Request. |
| General Status | USINT | One of the General Status codes listed in Appendix B Error Codes. If a routing error occurred, it shall be limited to the values indicated as Detected By Router devices in section 3-5.5, Connection Manager Object Instance Error Codes. |
| Size of Additional Status | USINT | Number of 16 bit words in Additional Status array. |
| Additional Status | Array of UINT | When returning an error from a target, which is a DeviceNet node, the Additional Status shall contain the 8 bit Additional Error Code from the target in the lower 8 bits and a zero (0) in the upper 8 bits. |
| Remaining Path Size | USINT | This field is only present with routing type errors and indicates the number of words in the original route path (*Route_Path* parameter of the Unconnected Send Request) as seen by the router that detects the error. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# 10-6 Examples

This section presents examples to further clarify the operation of the Connection Manager Object and the Unconnected Send service.

## 10-6.1 Using the Unconnected Send Service for Explicit Messaging – Single Hop

Assume that the *Requesting Device* wants to execute a Get_Attribute_Single on Class #1/Instance #2/Attribute ID #3 in the *Target Device*.

**Figure 10-6.1 Single Hop Unconnected Send (Explicit Message Request/Response)**

The table below presents the contents of the Service Data field for the Unconnected Send request which is used to execute the Get_Attribute_Single example noted above (all values are in hex).

**Table 10-6.1 Single Hop Unconnected Send Request Service Data**

| Bytes | Meaning |
|---|---|
| 01 00 | The Requesting Device generated Transaction_Id field. |
| pp | Priority/Tick Time field. |
| tt | Connection Timeout Ticks field. |
| 0C 00 | Message_Size field = 12 bytes. Note that the Request_Data field is empty. |
| 0E | Service field = Get_Attribute_Single. |
| 05 | Request_Path_Size field indicates that there are 5 words in the path field. |
| 21 00 01 00<br>25 00 02 00<br>30 03 | Path field specifying 16 bit Class Id = 1, 16 bit Instance ID = 2, 8 bit Attribute Id = 3. Note the presence of the pad bytes immediately following the 0x21 and 0x25 bytes. This path could have also been formatted as follows: 20 01 24 02 30 03 which makes use of 8 bit values to convey the Class and Instance ID data. In the 8-bit case, the pad bytes are not required. |
| 09 | Route_Path_Size field indicates that there are 9 words in the route path. |
| 00 | Reserved byte. |
| 13 0F 31 33<br>30 2E 31 35<br>31 2E 31 33<br>37 2E 31 30<br>35 00 | Contents of the Route_Path field. This field specifies the routing information. These bytes indicate that the request is to be delivered out Port #3 and to IP Address 130.151.137.105. This path encoding uses the Extended Link Address format for the IP address. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Step 1 – Request delivered to the DeviceNet to EtherNet/IP Router**

The first step in the process calls for the Requesting Device to establish an Explicit Messaging Connection with the DeviceNet to EtherNet/IP Router and issue the Unconnected Send request. The table below presents the entire Unconnected Send request. Assume that the Message Body Format established for the Explicit Messaging Connection is DeviceNet 8/8.

**Table 10-6.2 Explicit Message Request on DeviceNet**

| Bytes | Meaning |
|---|---|
| 07 | Frag = 0[1], Transaction ID = 0, Destination MACID = 7. |
| 52 | Service = 52. In this context, this specifies the Unconnected Send service. |
| 06 | Class ID of the Connection Manager Object Class. |
| 01 | The Instance ID Field specifying instance 1 of the Connection Manager. |
| 01 00 pp tt 0C 00<br>0E 05 21 00 01 00<br>25 00 02 00 30 03<br>09 00 13 0F 31 33<br>30 2E 31 35 31 2E<br>31 33 37 2E 31 30<br>35 00 | Unconnected Send Service Request Data. This is the previously described Unconnected Send service data describe above with the Transaction_Id field as the first UINT value (01 00 for this example). |

Table Footnotes

1. This request would have to be fragmented to deliver it to the router. Fragmentation is not illustrated in this example.

The Unconnected Send request has now been delivered to the DeviceNet to EtherNet/IP Router via an Explicit Messaging Connection that exists between the Requesting Device and the DeviceNet to EtherNet/IP Router.

**Step 2 – Router delivers the request to the Target Device**

When the Connection Manager Object within the DeviceNet Router receives the Unconnected Send request it examines the contents of the *Route_Path* field. In this case, the contents are "13 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30 35 00". This indicates that the next device in the hop is on Port #3 and its Node (IP) Address is 130.151.137.105. Additionally, since there is only a single Port/Node Address pair specified in the *Route_Path*, the request has reached its destination network. When a DeviceNet Router receives an Unconnected Send Request that DOES NOT need to be forwarded through more routers, the following steps are taken:

- The DeviceNet Router executes the timing related logic associated with the *Priority/Tick Time* and *Connection Timeout Ticks* fields. If a timeout is detected, then a successful Unconnected Send response that specifies a Routing Error is returned to the Requesting Device.
- The DeviceNet Router extracts the actual transaction from the Message Request portion of the Service Data and delivers the Explicit Messaging Request to the Target Device. The actual method of delivering the explicit message is dependent on the link type.

In this example, the message request inside the Unconnected Send indicates a Get_Attribute_Single to Class #1, Instance #2, Attribute #3. This approach has no effect on the Target Device. The Target Device does not even realize it has just responded to a request that originated on a remote DeviceNet.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Step 3 – Target returns the Explicit Message response to the Router**

The Target Device processes the explicit message and returns a response to the Router. When the Get_Attribute_Single Response is received by the DeviceNet Router it generates the Unconnected Send Response and internally delivers it to the Explicit Messaging Connection across which the Unconnected Send Request was originally received.

**Step 4 – Router returns the Unconnected Response to the Requesting Device**

The DeviceNet Router then delivers the Unconnected Send Response to the original Requesting Device.

Assume that the Target Device returns a successful response to the Get_Attribute_Single and the requested attribute is a UINT whose value is 0x1234. The text below presents the contents of the CAN Data Field associated with the Unconnected Send Response returned to the Requesting Device:

**Table 10-6.3 Successful Explicit Message Response on DeviceNet**

| Bytes | Meaning |
|---|---|
| 05 | Frag = 0, Transaction ID = 0, Destination MAC = 5. |
| 8E | Service = 8E. In this context, this specifies a response to the Unconnected Send service. |
| 01 00 | The Transaction_Id field echoed back in the response. |
| 00 | The General Status field. The value zero (0) indicates that there were no routing errors. |
| 00 | The Success status has no additional status. |
| 34 12 | The Target Device's Explicit Messaging Response Service Data field. In this case, the value 0x1234 was returned in the Get_Attribute_Single response. |

Now assume that the Target Device returned an Error Response whose General Error Code was set to the value "2" and whose Additional Error Code field was set to the value "5". The text below presents the contents of the CAN Data Field associated with the Unconnected Send Response returned to the Requesting Device. The absence of the *Remaining Path Size* field indicates that this error (Error Code = 2) was returned from the Target Device, not a router.

**Table 10-6.4 Unsuccessful Explicit Message Response on DeviceNet**

| Bytes | Meaning |
|---|---|
| 05 | Frag = 0, Transaction ID = 0, Destination MAC = 5. |
| 8E | Service = 0x8E. In this context, this specifies a response to the Unconnected Send service. |
| 01 00 | The Transaction_Id field echoed back in the response. |
| 02 | The General Status field. The non-zero value indicates there was an error. |
| 01 | The Size of Additional Status field. This indicates that the Target Device returned 16 bits of additional status. |
| 05 00 | This indicates that the Target Device returned an Additional Error Code of 0x05. |

Finally, assume that the DeviceNet Router was unable to establish an Explicit Messaging Connection with the Target Device. In this case a Routing Error has occurred (the request was not delivered to the Target Device). The error code and extended status that most closely conveys the routing error is: General Code = 0x01, Extended Code = 0x0204 – *Unconnected Send timed out waiting for a response* (Note that error handling is dealt with in more detail in the Event/Action Matrix in 10-4). The text below presents the contents of the CAN Data Field associated with the Unconnected Send Response returned to the Requesting Device:


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 10-6.5 Routing Failure Explicit Message Response on DeviceNet**

| Bytes | Meaning |
|---|---|
| 05 | Frag = 0, Transaction ID = 0, Destination MAC = 5. |
| D2 | Service = D2. In this context, this specifies a response to the Unconnected Send service. |
| 01 00 | The Transaction_Id field echoed back in the response. |
| 01 | The General Status field. This indicates that a routing error was detected. |
| 01 | The Size of Additional Status field. This indicates that there is 16 bits of additional status. |
| 04 02 | The Additional Status field. There is a single UINT whose value is 0x0204. |
| 00 | The *Remaining PathSize* field. The value zero (0) indicates that an error was encountered during the attempt to establish communications with the Remote Target Device versus another DeviceNet Router. |

## 10-6.2 Using the Unconnected Send Service For Explicit Messaging – Multiple Hop

Assume that the Requesting Device wants to execute a Set_Attribute_Single on Class #1/Instance #2/Attribute ID #3 in the Target Device. Assume that the data being sent to this attribute is a byte array whose value is 01 02 03 … 09. Notice that this transaction will flow through two DeviceNet Routers.

**Figure 10-6.2 Multiple Hop Unconnected Send**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Step 1 – Request delivered to DeviceNet to DeviceNet Router #1**

The first step in the process calls for the Requesting Device to establish an Explicit Messaging Connection with DeviceNet to DeviceNet Router #1 and issue the Unconnected Send request. This is identical to Step #1 in the previous example. The text below presents the contents of the Unconnected Send request's Service Data field, which is used to execute the Set_Attribute_Single noted above (all values are in hex).

**Table 10-6.6 Multiple Hop Unconnected Send Request Service Data (First Hop)**

| Bytes | Meaning |
|---|---|
| 01 00 | The Requesting Device generated Transaction_Id field. |
| 07 | Priority/Tick Time field (Time Tick = 128 ms). |
| 0C | Connection Timeout Ticks field (12 Ticks, which results in 1536 ms timeout). |
| 11 00 | Message_Size field = 17 bytes. Note that this means that a pad byte will be inserted between the Request_Data field and the Route_Path_Size field. |
| 10 | Service field = Set_Attribute_Single. |
| 03 | Request_Path_Size field indicates that there are 3 words in the request path. |
| 20 01 24 02 30 03 | Request_Path field specifying 8 bit Class Id = 1, 8 bit Instance ID = 2, 8 bit Attribute Id = 3. |
| 01 02 03 04 05 06 07 08 09 | Request_Data field specifying the attribute data associated with the Set_Attribute_Single request. |
| 00 | Pad byte (required due to the odd length). |
| 02 | Route_Size field indicates that there are 2 words in the Route_Path. |
| 00 | Reserved byte. |
| 02 09 04 06 | Contents of the Route_Path field specifying the routing information. These bytes indicate that the request is to be delivered out Port #2 and to MAC ID 9 and then out Port #4 to the Target Device at MAC ID 6. |

**Step 2 – Request delivered to DeviceNet to DeviceNet Router #2**

When DeviceNet Router #1 receives the Unconnected Send request with this Service Data field it detects that there are more routers to go through before the Target Device can be reached. In this case, DeviceNet Router #1 executes the following steps:

- Executes the timing related logic associated with the *Priority/Tick Time* and *Connection Timeout Ticks* fields. If a timeout is detected, then a successful Unconnected Send response, which specifies a Routing Error, is returned to the Requesting Device.
- Strips its portion of the routing information from the Unconnected Send request's *Route_Path* field and forwards the Unconnected Send Request accordingly. In this example, the "02 09" bytes constitute the routing information pertinent to DeviceNet Router #1 (next hop indicator). These bytes indicate that the Unconnected Send Request should be sent out Port #2 and to MAC ID #9 on that subnet.
- Establishes an Explicit Messaging Connection with DeviceNet Router #2 (if necessary) and sends the remaining data in the Unconnected Send service the second router.

The Service Data field of the Unconnected Send Request that DeviceNet Router #1 forwards to DeviceNet Router #2 is presented below. In this example, DeviceNet Router #1 is making use of the Transaction_Id field for internal transaction management purposes and that the value "1" which was received with the request is already in use. In this case the DeviceNet Router #1 is free to allocate an unused value (assume the value "4") and insert it into the packet it forwards.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table 10-6.7 Multiple Hop Unconnected Send Request Service Data (Second Hop)**

| Bytes | Meaning |
|---|---|
| 04 00 | The Transaction_Id field inserted by the router. |
| 07 | Priority/Tick Time field (Time Tick = 128 ms). |
| 08 | Connection Timeout Ticks field (8 Ticks, which results in 1024 ms timeout, 512 ms less than was originally received). |
| 11 00 | Message_Size field = 17 bytes. |
| 10 | Service field = Set_Attribute_Single. |
| 03 | Request_Path_Size field indicates that there are 3 words in the request path. |
| 20 01 24 02 30 03 | Request_Path field specifying 8 bit Class Id = 1, 8 bit Instance ID = 2, 8 bit Attribute Id = 3. |
| 01 02 03 04 05 06 07 08 09 | Request_Data field specifying the attribute data associated with the Set_Attribute_Single request. |
| 00 | Pad byte. |
| 01 | Route_Size field indicates that there is 1 word in the Route_Path. |
| 00 | Reserved byte. |
| 04 06 | Route_Path indicating that the request is to be delivered out Port #4 to the Target Device at MAC ID 6. Note that DeviceNet Router #1 has *stripped* its routing information from the packet. |

**Step 3 – Request delivered to Remote Target Device**

When DeviceNet Router #2 receives this Service Data field in the Unconnected Send Request it behaves identical to the DeviceNet Router in the first example. Specifically, DeviceNet Router #2 executes the following steps:

- Executes the timing related logic associated with the *Priority/Tick Time* and *Connection Timeout Ticks* fields. If a timeout is detected, then a successful Unconnected Send response that specifies a Routing Error is returned to the Requesting Device.
- Since there is no additional path routing, the device at MAC ID 6 is the Remote Target Device. The DeviceNet Router establishes an Explicit Messaging Connection with that device (if necessary), extracts the actual transaction from the Message Request portion of the Service Data, and delivers the Explicit Messaging Request to the Target Device.

The response returns back to the Requesting Device with each DeviceNet Router utilizing the Transaction_Id field to facilitate internal transaction management. If an error was detected at any point in the process, then the appropriate error information would be returned in the successful Unconnected Send Response.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 10-6.3 Using the Forward Open Service to Open an I/O Messaging Connection

Assume that the *Requesting Device* wants to make an I/O connection to an I/O Assembly. This data for this Assembly is identified as Class #4/Instance #2/Attribute ID #3 in the *Target Device*.

**Figure 10-6.3 Forward Open to Establish I/O Connection**

The table below presents the contents of the Service Data field for the Forward Open request which is used to execute the connection establishment example noted above (all values are in hex).

**Table 10-6.8 Forward Open Request Service Data**

| Byte Value | Meaning |
|---|---|
| pp | Priority/Tick Time field. |
| tt | Connection Timeout Ticks field. |
| 45 00 00 00 | O_to_T Connection ID – Chosen by originating node (CAN ID 0x045 = Node 5, Group 1, Msg ID 1) |
| FF FF FF FF | T_to_O Connection ID – Chosen by target node |
| 0C 00 | Connection Serial Number |
| FF FF | Originator Vendor ID (Vendor ID = 65535) |
| 00 01 02 03 | Originator Serial Number (Serial Number = 0x03020100) |
| 00 | Connection Timeout Multiplier |
| 00 00 00 | Reserved |
| 20 A1 07 00 | O_to_T RPI (0x7A120 = 500 ms) |
| | O_to_T Connection Parameters |
| 20 A1 07 00 | T_to_O RPI (0x7A120 = 500 ms) |
| | T_to_O Connection Parameters |
| 82 | Transport Type / Trigger (Class 2 Cyclic Server) |
| 0E | Connection_Path_Size field indicates that there are 14 words in the connection path. |
| 13 0F 31 33<br>30 2E 31 35<br>31 2E 31 33<br>37 2E 31 30<br>35 21 00 04<br>00 25 00 02<br>00 30 03 00 | Contents of the Connection_Path field. This field specifies the routing/connection information. These bytes indicate that the request is to be delivered out Port #3 and to IP Address 130.151.137.105. This path encoding uses the Extended Link Address format for the IP address. It further indicates that the application being connected to by specifying 16 bit Class ID = 4, 16 bit Instance ID = 2, 8 bit Attribute ID = 3. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Step 1 – Request delivered to the DeviceNet to EtherNet/IP Router**

The first step in the process calls for the Requesting Device to establish an Explicit Messaging Connection with the DeviceNet to EtherNet/IP Router and issue the Forward Open request. The table below presents the entire Forward Open request. Assume that the Message Body Format established for the Explicit Messaging Connection is DeviceNet 8/8.

**Table 10-6.9 Forward Open Request on DeviceNet**

| Bytes | Meaning |
|---|---|
| 07 | Frag = 0[1], Transaction ID = 0, Destination MACID = 7. |
| 54 | Service = 54. In this context, this specifies the Forward Open service. |
| 06 | Class ID of the Connection Manager Object Class. |
| 01 | The Instance ID Field specifying instance 1 of the Connection Manager. |
| 01 00 pp tt 0C 00 0E<br>05 21 00 04 00 25 00<br>02 00 30 03 09 00 13<br>0F 31 33 30 2E 31 35<br>31 2E 31 33 37 2E 31<br>30 35 00 | Forward Open Service Request Data. This is the previously described Forward Open service data described above. |

Table Footnotes

1. This request would have to be fragmented on DeviceNet to deliver it to the router. DeviceNet fragmentation is not illustrated in this example.

The Forward Open request has now been delivered to the DeviceNet Router via an Explicit Messaging Connection that exists between the Originating Device and the DeviceNet Router.

**Step 2 – Router creates a CIP Bridged connection**

Upon receipt of a Forward Open to the Connection Manager object, the router shall (if it has the resources available) create a CIP Bridged connection (Class Code 0x05) and place it in the Configuring state. The remaining attributes are set with either the value received in the service request or the appropriate default values. If resources are unavailable, an error response is returned.

**Step 3 – Router creates connection with the Target Device**

After the Connection Manager Object within the DeviceNet Router processes the Forward Open request it examines the contents of the Connection_Path field. In this case, the contents are “13 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30 35 21 00 04 00 25 00 02 00 30 03 00”. This indicates that the next device in the hop is on Port #3 and its Node (IP) Address is 130.151.137.105. Additionally, since there is only a single Port/Node Address pair specified in the Connection_Path field, the request has reached its destination network. When a DeviceNet Router receives a Forward Open Request that DOES NOT need to be forwarded through more routers, the following steps are taken:

- The DeviceNet Router executes the timing related logic associated with the Priority/Tick Time and Connection Timeout Ticks fields. If a timeout is detected, then a successful Forward Open response that specifies a Routing Error is returned to the Requesting Device.
- The DeviceNet Router creates an I/O connection with the Target Device. Once the I/O connection is created, the connection attributes are set, and the connection is applied. (Note that if the connection path indicates the Message Router object, then the router opens an Explicit Messaging connection using the UCMM.)

In the example above, the connection path inside the Forward Open indicates an I/O connection to Class #4, Instance #2, Attribute #3.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Step 4 – Router returns the Forward Open Response to the Requesting Device**

The DeviceNet Router then delivers the Forward Open Response to the original Requesting Device. This response will contain the actual packet rates and the connection identifiers to be used.

The service data field for a Forward Open response is shown below.

**Table 10-6.10 Forward Open Response Service Data**

| Byte Value | Meaning |
|---|---|
| 45 00 00 00 | O_to_T Connection ID – Chosen by originating node (CAN ID 0x045 = Node 5, Group 1, Msg ID 1) |
| 07 01 00 00 | T_to_O Connection ID – Chosen by target node (CAN ID 0x107 = Node 7, Group 1, Msg ID 4) |
| 0C 00 | Connection Serial Number |
| FF FF | Originator Vendor ID (Vendor ID = 65535) |
| 00 01 02 03 | Originator Serial Number (Serial Number = 0x03020100) |
| 00 | Connection Timeout Multiplier |
| 00 00 00 | Reserved |
| 20 A1 07 00 | O_to_T API (0x7A120 = 500 ms) |
| 20 A1 07 00 | T_to_O API (0x7A120 = 500 ms) |
| 00 | Application reply size (no application reply data) |
| 00 | Reserved |

Assume that the connection to the Target Device was successful. The text below presents the contents of the CAN Data Field associated with the Forward Open Response returned to the Requesting Device:

**Table 10-6.11 Forward Open Response on DeviceNet**

| Bytes | Meaning |
|---|---|
| 07 | Frag = 0, Transaction ID = 0, Destination MAC = 7. |
| D4 | Service = D4. In this context, this specifies a response to the Forward Open service. |
| 00 | The General Status field. The value zero (0) indicates that there were no routing errors. |
| 00 | Reserved byte. |
| 45 00 00 00<br>07 01 00 00<br>0C 00 FF FF<br>00 01 02 03<br>00 00 00 00<br>20 A1 07 00<br>20 A1 07 00<br>00 00 | Forward Open Service Response Data. This is the previously described Forward Open service data described above. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## 10-6.4 Using the Forward Close Service to Close an I/O Messaging Connection

The table below presents the contents of the Service Data field for the Forward Close request to delete the connection, which was established in the example above (all values are in hex).

**Table 10-6.12 Forward Close Request Service Data**

| Byte Value | Meaning |
|---|---|
| pp | Priority/Tick Time field. |
| tt | Connection Timeout Ticks field. |
| 0C 00 | Connection Serial Number |
| FF FF | Originator Vendor ID (Vendor ID = 65535) |
| 00 01 02 03 | Originator Serial Number (Serial Number = 0x03020100) |
| 0E | Connection_Path_Size field indicates that there are 14 words in the connection path. |
| 00 | Reserved |
| 13 0F 31 33<br>30 2E 31 35<br>31 2E 31 33<br>37 2E 31 30<br>35 21 00 04<br>00 25 00 02<br>00 30 03 00 | Contents of the Connection_Path field. This field specifies the routing/connection information. These bytes indicate that the request is to be delivered out Port #3 and to IP Address 130.151.137.105. This path encoding uses the Extended Link Address format for the IP address. It further indicates the application being disconnected from by specifying 16 bit Class ID = 4, 16 bit Instance ID = 2, 8 bit Attribute ID = 3. |

**Step 1 – Request delivered to the DeviceNet to EtherNet/IP Router**

The first step in the process calls for the Requesting Device to establish an Explicit Messaging Connection with the DeviceNet to EtherNet/IP Router (if one does not exist) and issue the Forward Close request. The table below presents the entire Forward Open request. Assume that the Message Body Format established for the Explicit Messaging Connection is DeviceNet 8/8.

**Table 10-6.13 Forward Close Request on DeviceNet**

| Bytes | Meaning |
|---|---|
| 07 | Frag = 0[1], Transaction ID = 0, Destination MACID = 7. |
| 5E | Service = 5E. In this context, this specifies the Forward Close service. |
| 06 | Class ID of the Connection Manager Object Class. |
| 01 | The Instance ID Field specifying instance 1 of the Connection Manager. |
| 01 00 pp tt 0C 00 0E<br>05 21 00 04 00 25 00<br>02 00 30 03 09 00 13<br>0F 31 33 30 2E 31 35<br>31 2E 31 33 37 2E 31<br>30 35 00 | Forward Close Service Request Data. This is the previously described Forward Close service data described above. |

Table Footnotes

1. This request would have to be fragmented to deliver it to the router. DeviceNet fragmentation is not illustrated in this example.

The Forward Close request has now been delivered to the DeviceNet Router via an Explicit Messaging Connection that exists between the Originating Device and the DeviceNet Router.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Step 2 – Router deletes the CIP Bridged connection with Target**

After the Connection Manager Object within the DeviceNet Router processes the Forward Close request it examines the contents of the *Connection_Path* field. In this case, the contents are "13 0F 31 33 30 2E 31 35 31 2E 31 33 37 2E 31 30 35 21 00 04 00 25 00 02 00 30 03 00". This indicates that the next device in the hop is on Port #3 and its Node (IP) Address is 130.151.137.105. Additionally, since there is only a single Port/Node Address pair specified in the *Connection_Path*, the request has reached its destination network. When a DeviceNet Router receives a Forward Close Request that DOES NOT need to be forwarded through more routers, the following steps are taken:

The DeviceNet Router executes the timing related logic associated with the *Priority/Tick Time* and *Connection Timeout Ticks* fields. If a timeout is detected, then a successful Forward Close response that specifies a Routing Error is returned to the Requesting Device.

The DeviceNet Router deletes the I/O or Explicit Messaging connection with the Target Device.

**Step 3 – Router returns the Forward Close Response to the Requesting Device and releases internal resources**

The DeviceNet Router then delivers the Forward Close Response to the original Requesting Device and releases all internal resources related to the connections on both the DeviceNet and EtherNet/IP subnets.

The service data field for a Forward Close response is shown below.

**Table 10-6.14 Forward Close Response Service Data**

| Byte Value | Meaning |
|---|---|
| `0C 00` | Connection Serial Number |
| `FF FF` | Originator Vendor ID (Vendor ID = 65535) |
| `00 01 02 03` | Originator Serial Number (Serial Number = 0x03020100) |
| `00` | Application reply size (no application reply data) |
| `00` | Reserved |

Assume that the connection to the Target Device was successful. The text below presents the contents of the CAN Data Field associated with the Forward Close Response returned to the Requesting Device:

**Table 10-6.15 Forward Close Response on DeviceNet**

| Bytes | Meaning |
|---|---|
| `07` | Frag = 0, Transaction ID = 0, Destination MAC = 7. |
| `DE` | Service = DE. In this context, this specifies a response to the Forward Close service. |
| `00` | The General Status field. The value zero (0) indicates that there were no routing errors. |
| `00` | Reserved byte. |
| `0C 00 FF FF`<br>`00 01 02 03`<br>`00 00` | Forward Close Service Response Data. This is the previously described Forward Close service data described above. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Volume 1: Common Industrial Protocol (CIP™) Specification

# Appendix A: Explicit Messaging Services


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Contents

A-1 Introduction ..... 5
A-2 Service Definitions ..... 5
A-3 CIP Common Services ..... 6
A-4 CIP Common Service Definitions ..... 7
A-4.1 Get_Attributes_All - Service Code: $01_{Hex}$ ..... 7
A-4.1.1 Service Requirements ..... 7
A-4.1.2 Request Service Data Field Parameters ..... 7
A-4.1.3 Success Response Service Data Field Parameters ..... 8
A-4.2 Set_Attributes_All - Service Code: $02_{Hex}$ ..... 9
A-4.2.1 Service Requirements ..... 9
A-4.2.2 Request Service Data Field Parameters ..... 9
A-4.2.3 Success Response Service Data Field Parameters ..... 10
A-4.3 Get_Attribute_List - Service Code: $03_{Hex}$ ..... 11
A-4.3.1 Service Requirements ..... 11
A-4.3.2 Request Service Data Field Parameters ..... 11
A-4.3.3 Success Response Service Data Field Parameters ..... 11
A-4.4 Set_Attribute_List - Service Code: $04_{Hex}$ ..... 12
A-4.4.1 Service Requirements ..... 12
A-4.4.2 Request Service Data Field Parameters ..... 12
A-4.4.3 Success Response Service Data Field Parameters ..... 12
A-4.5 Reset - Service Code: $05_{hex}$ ..... 13
A-4.5.1 Service Requirements ..... 13
A-4.5.2 Request Service Data Field Parameters ..... 13
A-4.5.3 Success Response Service Data Field Parameters ..... 13
A-4.6 Start - Service Code: $06_{Hex}$ ..... 14
A-4.6.1 Service Requirements ..... 14
A-4.6.2 Request Service Data Field Parameters ..... 14
A-4.6.3 Success Response Service Data Field Parameters ..... 14
A-4.7 Stop - Service Code: $07_{Hex}$ ..... 15
A-4.7.1 Service Requirements ..... 15
A-4.7.2 Request Service Data Field Parameters ..... 15
A-4.7.3 Success Response Service Data Field Parameters ..... 15
A-4.8 Create - Service Code: $08_{Hex}$ ..... 16
A-4.8.1 Service Requirements ..... 16
A-4.8.2 Request Service Data Field Parameters ..... 16
A-4.8.3 Success Response Service Data Field Parameters ..... 16
A-4.9 Delete - Service Code: $09_{Hex}$ ..... 17
A-4.9.1 Service Requirements ..... 17
A-4.9.2 Request Service Data Field Parameters ..... 17
A-4.9.3 Success Response Service Data Field Parameters ..... 17
A-4.10 Multiple_Service_Packet - Service Code: $0A_{Hex}$ ..... 18
A-4.10.1 Service Requirements ..... 18
A-4.10.2 Request Service Data Field Parameters ..... 18
A-4.10.3 Success Response Service Data Field Parameters ..... 18
A-4.11 Apply_Attributes - Service Code: $0D_{Hex}$ ..... 20
A-4.11.1 Service Requirements ..... 20
A-4.11.2 Request Service Data Field Parameters ..... 20
A-4.11.3 Success Response Service Data Field Parameters ..... 20
A-4.12 Get_Attribute_Single - Service Code: $0E_{Hex}$ ..... 21
A-4.12.1 Service Requirements ..... 21
A-4.12.2 Request Service Data Field Parameters ..... 21
A-4.12.3 Success Response Service Data Field Parameters ..... 21
A-4.13 Set_Attribute_Single - Service Code: $10_{Hex}$ ..... 22


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

A-4.13.1 Service Requirements ..... 22
A-4.13.2 Request Service Data Field Parameters ..... 22
A-4.13.3 Success Response Service Data Field Parameters ..... 22
A-4.14 Find_Next_Object_Instance - Service Code: $11_{Hex}$ ..... 23
A-4.14.1 Service Requirements ..... 23
A-4.14.2 Request Service Data Field Parameters ..... 24
A-4.14.3 Success Response Service Data Field Parameters ..... 24
A-4.15 Restore - Service Code: $15_{Hex}$ ..... 25
A-4.15.1 Service Requirements ..... 25
A-4.15.2 Request Service Data Field Parameters ..... 25
A-4.15.3 Success Response Service Data Field Parameters ..... 25
A-4.16 Save - Service Code: $16_{Hex}$ ..... 26
A-4.16.1 Service Requirements ..... 26
A-4.16.2 Request Service Data Field Parameters ..... 26
A-4.16.3 Success Response Service Data Field Parameters ..... 26
A-4.17 No Operation (NOP) - Service Code: $17_{Hex}$ ..... 27
A-4.17.1 Required Behavior ..... 27
A-4.17.2 Request Service Data Field Parameters ..... 27
A-4.17.3 Success Response Service Data Field Parameters ..... 27
A-4.18 Get_Member - Service Code: $18_{Hex}$ ..... 28
A-4.18.1 Service Requirements ..... 28
A-4.18.2 Request Service Data Field Parameters ..... 28
A-4.18.3 Success Response Service Data Field Parameters ..... 28
A-4.19 Set_Member - Service Code: $19_{Hex}$ ..... 29
A-4.19.1 Service Requirements ..... 29
A-4.19.2 Request Service Data Field Parameters ..... 29
A-4.19.3 Success Response Service Data Field Parameters ..... 29
A-4.20 Insert_Member - Service Code: $1A_{Hex}$ ..... 30
A-4.20.1 Service Requirements ..... 30
A-4.20.2 Request Service Data Field Parameters ..... 30
A-4.20.3 Success Response Service Data Field Parameters ..... 30
A-4.21 Remove_Member - Service Code: $1B_{Hex}$ ..... 31
A-4.21.1 Service Requirements ..... 31
A-4.21.2 Request Service Data Field Parameters ..... 31
A-4.21.3 Success Response Service Data Field Parameters ..... 31
A-4.22 GroupSync - Service Code: $1C_{Hex}$ ..... 32
A-4.22.1 Service Requirements ..... 32
A-4.22.2 Request Service Data Field Parameters ..... 32
A-4.22.3 Response Service Data Field Parameters ..... 32
A-5 Member Service Protocols ..... 33
A-5.1 Member ID/EX Description ..... 33
A-5.2 Member Request Message - Basic ..... 34
A-5.3 Member Request Message - Extended ..... 35
A-5.3.1 Multiple Sequential Members - Extended Protocol ..... 36
A-5.3.2 International String Selection– Extended Protocol ..... 37
A-6 CIP Encoding Examples ..... 39
A-6.1 Example Object Class Definitions ..... 39
A-6.2 Encoding Examples ..... 40
A-6.2.1 Get_Attributes_All ..... 41
A-6.2.2 Set_Attributes_All ..... 42
A-6.2.3 Reset ..... 43
A-6.2.4 Start ..... 44
A-6.2.5 Stop ..... 45
A-6.2.6 Create ..... 46
A-6.2.7 Delete ..... 47
A-6.2.8 Apply_Attributes ..... 48


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

A-6.2.9 Get_Attribute_Single ..... 49
A-6.2.10 Set_Attribute_Single ..... 50
A-6.2.11 Find_Next_Object_Instance ..... 51
A-6.2.12 Restore ..... 52
A-6.2.13 Save ..... 53
A-6.2.14 No Operation ..... 54
A-6.2.15 Error Response ..... 55
A-6.2.16 Get_Member ..... 56
A-6.2.17 Set_Member ..... 57
A-6.2.18 Insert_Member ..... 58
A-6.2.19 Remove_Member ..... 59


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# A-1 Introduction

This appendix contains information about Explicit Messaging services. The CIP *Common Services* are defined and examples illustrating the encoding of CIP Common Services are provided. CIP Common Services are those services whose request/response parameters and required behaviors are defined in this document.

# A-2 Service Definitions

For any CIP service to be considered "fully defined," its definition must include the following information:

- A brief functional definition explaining what the service does (why an object would request the service)
- Behaviors associated with the service
- Parameters which are placed in the Service Data Field of an Explicit Request Message, including:
    - Data type
    - Description
- Parameters which are placed in the Service Data Field of an Explicit Response Message, including:
    - Data type
    - Description


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# A-3 CIP Common Services

The codes and names of the CIP Common Services are listed below.

**Table A-3.1 CIP Service Codes and Names**

| Service Code (in hex) | Section Number | Service Name |
|---|---|---|
| 00 | n/a | Reserved for future use |
| 01 | A-4.1 | Get_Attributes_All |
| 02 | A-4.2 | Set_Attributes_All |
| 03 | A-4.3 | Get_Attribute_List |
| 04 | A-4.4 | Set_Attribute_List |
| 05 | A-4.5 | Reset |
| 06 | A-4.6 | Start |
| 07 | A-4.7 | Stop |
| 08 | A-4.8 | Create |
| 09 | A-4.9 | Delete |
| 0A | A-4.10 | Multiple_Service_Packet |
| 0B-0C | n/a | Reserved for future use |
| 0D | A-4.11 | Apply_Attributes |
| 0E | A-4.12 | Get_Attribute_Single |
| 0F | n/a | Reserved for future use |
| 10 | A-4.13 | Set_Attribute_Single |
| 11 | A-4.14 | Find_Next_Object_Instance |
| 12 - 13 | n/a | Reserved for future use |
| 14 | n/a | Error Response (used by DeviceNet only) |
| 15 | A-4.15 | Restore |
| 16 | A-4.16 | Save |
| 17 | A-4.17 | No Operation (NOP) |
| 18 | A-4.18 | Get_Member |
| 19 | A-4.19 | Set_Member |
| 1A | A-4.20 | Insert_Member |
| 1B | A-4.21 | Remove_Member |
| 1C | A-4.22 | GroupSync |
| 1D-31 | n/a | Reserved for additional Common Services |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# A-4 CIP Common Service Definitions

This section provides a general description of the CIP Common Services. Objects Classes/Instances that utilize these Services must provide a detailed description of their specific use.

## A-4.1 Get_Attributes_All - Service Code: $01_{Hex}$

Returns the contents of the instance or class attributes defined in the object definition.

### A-4.1.1 Service Requirements

The following list details requirements associated with the Get_Attributes_All service:

1. The structure of the information in the successful response's Service Data Field adheres to the Get_Attributes_All response structure defined by the object or class. Support of this service requires the Object and/or Class to provide a detailed definition of the format of the data sent in the response message.
2. If the request is successfully serviced, then a Get_Attributes_All Response is returned. The Service Data Field of the response contains the attribute data. If an error is detected, then an Error Response is returned
3. A device is only required to include data in the response data array up to the last implemented attribute.
4. If the length of data in the response is less than documented in the object definition and the last part of the response data does not include a complete attribute, the response data is in error and the client shall discard it.
5. If the length of data in the response is less than documented in the object definition, the client assumes default values for the missing attributes.
6. If the length of data in the response is greater than the expected amount, the client ignores any data in excess of what was expected.

**Important:** Insert default values for all unsupported attributes present in the response byte array.

**Important:** Insert a zero count and no array elements for attributes that specify a count of array elements in another attribute, if the attribute pair is not supported. An example of this type of attribute is the Discrete Input Point Object (class code 08hex), instance attribute 1, Number of Attributes.

**Important:** Insert a zero count for attributes that specify a count followed by an array of entries, if the attribute is not supported. An example of this type of attribute is class attribute 4, Optional attribute list.

Refer to Chapter 4 for rules for how the Get_Attributes_All response data is defined.

### A-4.1.2 Request Service Data Field Parameters

None.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-4.1.3 Success Response Service Data Field Parameters

The following information is specified within the Service Data Field of a successful response to a Get_Attributes_All request

**Table A-4.1 Service Data for Get_Attributes_All Success Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Attribute Values | Class/Object-specific | A stream of information containing the values of each attribute that the Class/Object defines for the response format. Classes/Objects which support this service must define the format of this parameter. See Section 4-5.1 for definition requirements. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.2 Set_Attributes_All - Service Code: $02_{Hex}$

Modifies the contents of the instance or class attributes defined in the object definition.

### A-4.2.1 Service Requirements

The following list details requirements associated with the Set_Attributes_All service:

1. The structure of the information in the request's Service Data Field adheres to the definition of the Set_Attributes_All request for the object or class. Support of this service requires the Object and/or Class to provide a detailed definition of the format of the data sent in the request message.
2. If the ability to modify an attribute changes based on the state of the object, the object definition must provide a detailed description of how this service is affected. For example, this service could only be supported in a state where all attributes are modifiable.
3. Attributes will be modified only if all attribute specific value verifications (e.g. range checks, etc.) are successful. The first attribute failing verification will be specified in the Additional Code parameter of the Error Response message.
4. If any other error is detected, then an Error Response is returned.
5. If all verification checks pass, then the attributes are modified and a Set_Attributes_All success response is returned.
6. If the length of data in the service is less than documented in the object definition and the last part of the request data does not include a complete attribute, the "Not enough data" (0x13) Error Response is returned.
7. If the length of data in the service data field is less than documented in the object definition, this indicates the attributes represented by the missing data are not supported by the client. Missing data shall result in no modification to the corresponding attributes.
8. If the length of data in the request is greater than the expected amount, the "Too much data" (0x15) Error Response is returned.
9. If a device's implementation of an object does not support one or more of the optional attributes contained in that object's Set_Attributes_All request format, then it shall either:
   a. Omit support for the service (if the object definition allows),
   b. Reject the request with an "Attribute Not Supported" (0x14) error, or
   c. Accept the request provided the value(s) contained in the request for the unimplemented attribute(s) are the Default Values defined for those attribute(s). If any of the values for the unimplemented attribute(s) are not the Default Value defined for those attribute(s), the request shall be rejected. An "Invalid Attribute Value" (0x09) error code is recommended for this case.

Refer to Chapter 4 for rules how the Set_Attributes_All service data is defined.

### A-4.2.2 Request Service Data Field Parameters

The following information is specified within the Service Data Field of a Set_Attributes_All request.

**Table A-4.2 Service Data for Set_Attributes_All Request**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Attribute Values | Class/Object–specific | A stream of information containing the values of each attribute that the Class/Object defines for the request format. Classes/Objects which support this service must define the format of this parameter. See Section 4-5.2 for definition requirements. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-4.2.3 Success Response Service Data Field Parameters

None.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.3 Get_Attribute_List - Service Code: $03_{Hex}$

The Get_Attribute_List service shall return the contents of the selected gettable attributes of the specified object class or instance.

### A-4.3.1 Service Requirements

The following list details requirements associated with the Get_Attribute_List service:

1. The attributes shall be specified using a list of attribute identifiers.
2. The number of attributes actually returned shall be reported within the response. If there is not enough space for an attribute's data in the response message, a partial response shall be returned.
3. Attribute data returned within partial response messages shall be complete. The client application can submit another Get_Attribute_List service request for the attribute data remaining from its original list.
4. The client application shall verify the value of the attribute count parameter in the response.
5. Status shall be reported with each individual attribute. Attribute data shall be retrieved and packed in the sequence specified in the request.
6. When "Attribute Status" is a value other than "Success" (0x00), the "Attribute Data" shall not be returned.

### A-4.3.2 Request Service Data Field Parameters

The following information is specified within the Service Data Field of a Get_Attribute_List request.

**Table A-4.3 Service Data for Get_Attribute_List Request**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Attribute_count | UINT | Number of attribute identifiers in the attribute list |
| Attribute_list | ARRAY of UINT | List of the attribute identifiers to get from the class or object |

### A-4.3.3 Success Response Service Data Field Parameters

The following information is specified within the Service Data Field of a successful Get_Attribute_List response.

**Table A-4.4 Service Data for Get_Attribute_List Success Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Attribute_count | UINT | Number of "Attribute response structures" returned |
| Attribute response structures | LIST of STRUCT: | A list of response structures |
| | UINT | "Attribute identifier", the number of the attribute identifier being returned |
| | UINT | "Attribute Status", the status of the attribute response (per appendix B) |
| | Object/Class specific attribute data | "Attribute Data", Attribute response data - exists when "Attribute Status" is the value "Success" (0x00). |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.4 Set_Attribute_List - Service Code: $04_{Hex}$

The Set_Attribute_List service shall set the contents of selected attributes of the specified object class or instance.

### A-4.4.1 Service Requirements

The following list details requirements associated with the Set_Attribute_List service:

1. The data for each individual attribute shall be placed into the request in its entirety.
2. Each set activity shall be performed in the order specified in the list.
3. Status shall be reported for each of the individual attributes in the response.
4. A server shall not attempt to set an attribute for which it can not return a response status.

### A-4.4.2 Request Service Data Field Parameters

The service data field of the Set_Attribute_List request shall be as specified in the following table.

**Table A-4.5 Service Data for Set_Attribute_List Request**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Attribute_count | UINT | Number of attribute identifiers in the attribute list |
| Attributes request structures | LIST of STRUCT: | List of structures specific to this service |
| | UINT | “Attribute Identifier” - Attribute identification number |
| | Object / class specific data | Related attribute data |

### A-4.4.3 Success Response Service Data Field Parameters

The following information is specified within the Service Data Field of a successful Set_Attribute_List response.

**Table A-4.6 Service Data for Set_Attribute_List Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Attribute_count | UINT | Number of attribute values being returned |
| Attribute response structures | LIST of STRUCT: | A list of response structures |
| | UINT | “Attribute identifier”, the number of the attribute identifier being returned |
| | UINT | “Attribute Status”, the status of the attribute response (per appendix B) |
| | Object/Class specific attribute data | “Attribute Data”, Attribute response data - may exist when “Attribute Status” is the value “Success” (0x00). |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.5 Reset - Service Code: $05_{hex}$

Invokes the *Reset* service of the specified Class/Object. Typically this would cause a transition to a default state or mode.

### A-4.5.1 Service Requirements

The following list details requirements associated with the Reset service:

1. If the execution of the Reset service request would place the object/device in a state that will enable it to respond to the requester, then the response is not returned until the service has completed. If the execution of the Reset service would place the object/device in a state in which it may not be able to respond, the object/device must respond prior to executing the Reset service.
2. If an error is detected, then an Error Response is returned. Otherwise, a successful Reset response is returned.

### A-4.5.2 Request Service Data Field Parameters

The following information is **OPTIONALLY** specified within the Service Data Field of a Reset request.

**Table A-4.7 Service Data for Reset Request**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object Specific Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |

### A-4.5.3 Success Response Service Data Field Parameters

The following information is **OPTIONALLY** specified within the Service Data Field of a successful response to a Reset request

**Table A-4.8 Service Data for Reset Success Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object Specific Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.6 Start - Service Code: $06_{Hex}$

Invokes the *Start* service of the specified Class/Object. Typically, this would place an object into a running state/mode.

### A-4.6.1 Service Requirements

The following list details requirements associated with the Start service:

1. Other than documenting the state machine associated with the object/class relative to this service, there are no special requirements defined by CIP.
2. If an error is detected, then an Error Response is returned. Otherwise, a successful Start response is returned.

### A-4.6.2 Request Service Data Field Parameters

The following information is **OPTIONALLY** specified within the Service Data Field of a Start request.

**Table A-4.9 Service Data for Start Request**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object Specific Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |

### A-4.6.3 Success Response Service Data Field Parameters

The following information is **OPTIONALLY** specified within the Service Data Field of a successful response to a Start request

**Table A-4.10 Service Data for Start Success Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object Specific Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.7 Stop - Service Code: $07_{Hex}$

Invokes the *Stop* service of the specified Class/Object. Typically, this would place an object into a stopped or idle state/mode.

### A-4.7.1 Service Requirements

The following list details requirements associated with the Stop service:

1. Other than documenting the state machine associated with the object/class relative to this service, there are no special requirements defined by CIP.
2. If an error is detected, then an Error Response is returned. Otherwise, a successful Stop response is returned.

### A-4.7.2 Request Service Data Field Parameters

The following information is **OPTIONALLY** specified within the Service Data Field of a Stop request.

**Table A-4.11 Service Data for Stop Request**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object Specific Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |

### A-4.7.3 Success Response Service Data Field Parameters

The following information is **OPTIONALLY** specified within the Service Data Field of a successful response to a Stop request

**Table A-4.12 Service Data for Stop Success Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object Specific Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.8 Create - Service Code: $08_{Hex}$

Results in the instantiation of a new object within the specified class.

### A-4.8.1 Service Requirements

The following list details requirements associated with the Create service:

1. The object instance is created and initialized in accordance with the object definition.
2. Any error will result in the object instance not being created.
3. If an error is detected, then an Error Response is returned. Otherwise, a successful Create response is returned.

### A-4.8.2 Request Service Data Field Parameters

The following information is **OPTIONALLY** specified within the Service Data Field of a Create request.

**Table A-4.13 Service Data for Create Request**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object Specific Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |

### A-4.8.3 Success Response Service Data Field Parameters

The following information is specified within the Service Data Field of a successful response to a Create request

**Table A-4.14 Service Data for Create Success Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Instance ID | UINT | The integer value assigned to identify the newly created object. If the object definition does not define the Create service response instance ID, then the Instance ID is specified within a 16-bit field. |
| Object Specific Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.9 Delete - Service Code: $09_{Hex}$

Deletes an object instance of the specified class.

### A-4.9.1 Service Requirements

The following list details requirements associated with the Delete service:

1. All resources are deallocated and returned to the system.
2. If an error is detected, then an Error Response is returned. Otherwise, a successful Delete response is returned.

### A-4.9.2 Request Service Data Field Parameters

The following information is **OPTIONALLY** specified within the Service Data Field of a Delete request.

**Table A-4.15 Service Data for Delete Request**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object Specific Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |

### A-4.9.3 Success Response Service Data Field Parameters

The following information is **OPTIONALLY** specified within the Service Data Field of a successful response to a Delete request

**Table A-4.16 Service Data for Delete Success Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object Specific Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.10 Multiple_Service_Packet - Service Code: $0A_{Hex}$

Performs a set of services as an autonomous sequence.

### A-4.10.1 Service Requirements

The following list details requirements associated with the Multiple_Service_Packet service:

1. Performs services as an autonomous sequence of services.
2. Performs services in the sequence supplied.
3. Performs all services, reporting individual responses for each one.
4. Packs responses into the response buffer in the sequence in which they were executed.
5. A response timeout must be implemented for those service requests that do not guarantee a response.
6. Each embedded service may return a success or failure, as indicted in the response structure. If one or more service requests result in an error this service shall return an error. The error code returned shall be $1E_{hex}$ (Embedded service error).

This service allows clients to submit a sequence of 'embedded' services in a single message packet. The object processing the Multiple_Service_Packet shall not perform any other service until the entire sequence of embedded services has been attempted.

The embedded services are formatted according to their definitions. Each embedded service may contain a path. If a path is present, the embedded service is passed to the Message Router for processing. If a path is not present, the object performing the Multiple_Service_Packet request shall perform the embedded service.

### A-4.10.2 Request Service Data Field Parameters

The following information is specified within the Service Data Field of a Multiple_Service_Packet request.

**Table A-4.17 Service Data for Multiple_Service_Packet Request**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Number of Services | UINT | Number of embedded services in the Service List. |
| Service Offsets | ARRAY of UINT | Array of byte offsets to the start of each embedded service in the Service List. |
| Service List | ARRAY of STRUCT of | Array of service request structures. The format of the service request follows the Message Router Request header defined in Chapter 2. |
| | USINT | Service code of the request. |
| | USINT | The number of 16 bit words in the Request_Path field (next element). |
| | Padded EPATH | This is an array of bytes whose contents convey the path of the request (Class ID, Instance ID, etc.) for this transaction. |
| | ARRAY of octet | Service specific data to be delivered in the Explicit Messaging Request. If no additional data is to be sent with the Explicit Messaging Request, then this array will be empty. |

### A-4.10.3 Success Response Service Data Field Parameters

The following information is specified within the Service Data Field of a successful response to a Multiple_Service_Packet request


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table A-4.18 Service Data for Multiple_Service_Packet Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Number of Responses | UINT | Number of embedded services responses in the Response List. |
| Response Offsets | ARRAY of UINT | Array of byte offsets to the start of each embedded service response in the Response List. |
| Response List | ARRAY of STRUCT of | Array of service response structures. The format of the service response follows the Message Router Response header defined in Chapter 2. |
| | USINT | Reply service code. |
| | USINT | Shall be zero. |
| | USINT | One of the General Status codes listed in Appendix B (Status Codes). |
| | USINT | Number of 16 bit words in Additional Status array. |
| | ARRAY of UINT | Additional status. |
| | ARRAY of octet | Response data from request or additional error data if General Status indicated an error. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.11 Apply_Attributes - Service Code: $0D_{Hex}$

Causes attribute values whose use is *pending* to become actively used.

### A-4.11.1 Service Requirements

The following list details requirements associated with the Apply_Attributes service:

1. Data for pending attributes must be verified before it is actually used.
2. If an error is detected, then an Error Response is returned. Otherwise, a successful Apply_Attributes response is returned.

### A-4.11.2 Request Service Data Field Parameters

The following information is **OPTIONALLY** specified within the Service Data Field of an Apply_Attributes request.

**Table A-4.19 Service Data for Apply_Attributes Request**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object Specific Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |

### A-4.11.3 Success Response Service Data Field Parameters

The following information is OPTIONALLY specified within the Service Data Field of a successful response to an Apply_Attributes request

**Table A-4.20 Service Data for Apply_Attributes Success Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object Specific Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.12 Get_Attribute_Single - Service Code: $0E_{Hex}$

Returns the contents of the specified attribute.

### A-4.12.1 Service Requirements

The following list details requirements associated with the Get_Attribute_Single service:

1. The service causes the class/object to return the contents of the specified attribute or sub-attribute specified by Bit Index, Indirect Bit Index, Array Index, Indirect Array Index, Structure Member or Structure Handle to the requester.
2. If an error is detected, then an Error Response is returned. Otherwise, a successful Get_Attribute_Single response is returned along with the requested attribute data.

### A-4.12.2 Request Service Data Field Parameters

There are no request service data parameters except as noted in the following paragraph.

DeviceNet and CompoNet support link specific explicit message formats (i.e. they do not support the Message Router Request Format specified in 2-4.1). Some of these formats (i.e. the Compact Message Type Request Format on CompoNet and the Message Body Formats 0-3 on DeviceNet) do not enable the attribute ID to be specified in the header. For those formats, the attribute ID is limited to a USINT and is the first parameter (byte) of the service data.

### A-4.12.3 Success Response Service Data Field Parameters

The following information is specified within the Service Data Field of a successful Get_Attribute_Single response.

**Table A-4.22 Service Data for Get_Attribute_Single Success Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Attribute Data | Object/class attribute –specific Struct | Contains the requested attribute data |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.13 Set_Attribute_Single - Service Code: $10_{Hex}$

Modifies an attribute value.

### A-4.13.1 Service Requirements

The following list details requirements associated with the Set_Attribute_Single service:

1. The attribute or sub-attribute specified by Bit Index, Indirect Bit Index, Array Index, Indirect Array Index, Structure Member or Structure Handle data is validated prior to the modification taking place.
2. If an error is detected, then an Error Response is returned. Otherwise a successful Set_Attribute_Single response is returned.

### A-4.13.2 Request Service Data Field Parameters

The following information is specified within the Service Data Field of a Set_Attribute_Single request.

**Table A-4.23 Service Data for Set_Attribute_Single Request**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Attribute ID[1] | USINT | Identifies the attribute to be modified |
| Attribute Data | Attribute specific | Contains the value to which the specified attribute is to be modified. |

Table Footnotes

1 This parameter is conditional; it is only allowed in certain DeviceNet and CompoNet cases. DeviceNet and CompoNet support link specific explicit message formats (i.e. they do not support the Message Router Request Format specified in 2-4.1). Some of these formats (i.e. the Compact Message Type Request Format on CompoNet and the Message Body Formats 0-3 on DeviceNet) do not enable the attribute ID to be specified in the header. For those formats, the attribute ID is limited to a USINT and is the first parameter (byte) of the service data.

### A-4.13.3 Success Response Service Data Field Parameters

The following information is OPTIONALLY specified within the Service Data Field of a successful response to a Set_Attribute_Single request

**Table A-4.24 Service Data for Set_Attribute_Single Success Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.14 Find_Next_Object_Instance - Service Code: $11_{Hex}$

This service is supported at the Class level only. It causes the specified Class to search for and return a list of Instance IDs associated with existing Object Instances. *Existing* Objects are those that are currently accessible from the CIP subnet.

### A-4.14.1 Service Requirements

The following list details requirements associated with the Find_Next_Object_Instance service:

1. The Class utilizes the value specified in the Instance ID of the request message to determine the starting point for the search as described below:
   - If the Instance ID in the request message is zero (0), then the Class starts with the numerically lowest Instance ID.
   - If the Instance ID in the request message is not zero (0), then the Class starts with the next Instance ID whose value is numerically greater than the specified Instance ID.
   - If the Instance ID in the request message is greater than or equal to the numerically highest Instance ID within the Class, then the value zero (0) is returned.
2. The Class returns a list of Instance IDs associated with **existing** Objects beginning at the starting point and continuing in ascending Instance ID value order.
3. The request specifies the maximum number of Instance ID values to be returned in the response. The responding Class can return any number of Instance IDs less than or equal to the maximum specified in the request.
4. The responding device returns Instance ID value zero (0) to indicate that the end of the list has been reached.
5. If an error is detected, then an Error Response is returned. Otherwise, a successful Find_Next_Object_Instance response is returned.

The following illustration provides a general example of this service. Specific encoding examples are presented in section A-6.

**Figure A-4.25 Example of Find_Next_Object_Instance Message Flow**

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-4.14.2 Request Service Data Field Parameters

The following information is specified within the Service Data Field of a Find_Next_Object_Instance request.

**Table A-4.26 Service Data for Find_Next_Object_Instance Request**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Maximum Returned Values | USINT | Indicates the **maximum** number of Instance ID values to be returned in the response message. |

### A-4.14.3 Success Response Service Data Field Parameters

The following information is specified within the Service Data Field of a successful Find_Next_Object_Instance response.

**Table A-4.27 Service Data for Find_Next_Object_Instance Success Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Number Of List Members | USINT | Contains the number of Instance IDs specified in this response message. |
| Instance ID List | Array of UINT | Contains the returned Instance ID List. The Instance IDs are returned in 16 bit integer fields. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.15 Restore - Service Code: $15_{Hex}$

Restores the contents of a class/object's attributes from a storage location accessible by the Save service. Attribute data is copied from a storage area to the *currently active* memory area used by the class/object.

### A-4.15.1 Service Requirements

The following list details requirements associated with the Restore service:

1. Attribute data must be verified before the copy from the "storage area" to the "actively used" area is performed.
2. If the ability to modify an attribute changes based on the state of the object, the object definition must provide a detailed description of how this service is effected. For example; this service could only be supported in a state where all attributes are modifiable. Alternatively, the object could ignore the data associated with a currently non-modifiable attribute.
3. Attributes will be modified only if all attribute specific value verifications (e.g. range checks, etc.) are successful. The first attribute failing verification will be specified in the Additional Code parameter of the Error Response message.
4. If any other error is detected, then an Error Response is returned.
5. If all verification checks pass, then the attributes are modified and a Restore success response is returned.

### A-4.15.2 Request Service Data Field Parameters

The following information is **OPTIONALLY** specified within the Service Data Field of a Restore request.

**Table A-4.28 Service Data for Restore Request**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object Specific Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |

### A-4.15.3 Success Response Service Data Field Parameters

The following information is **OPTIONALLY** specified within the Service Data Field of a successful response to a Restore request

**Table A-4.29 Service Data for Restore Success Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object Specific Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.16 Save - Service Code: $16_{Hex}$

Copies the contents of an class/object's attributes to a location accessible by the Restore service.

### A-4.16.1 Service Requirements

The following list details requirements associated with the Save service:

1. The service will report success only after the copy has been completed and verified.
2. If an error is detected, then an Error Response is returned. Otherwise, a successful Save response is returned.

### A-4.16.2 Request Service Data Field Parameters

The following information is **OPTIONALLY** specified within the Service Data Field of a Save request.

**Table A-4.30 Service Data for Save Request**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object Specific Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |

### A-4.16.3 Success Response Service Data Field Parameters

The following information is **OPTIONALLY** specified within the Service Data Field of a successful response to a Save request

**Table A-4.31 Service Data for Save Success Response**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object Specific Data | Object/class service–specific STRUCT | Contains Class/Instance specific parameters. If a Class/Instance utilizes this field, then the Class/Instance definition must specify its format. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.17 No Operation (NOP) - Service Code: $17_{Hex}$

This service merely causes the receiving object to return a *No Operation* response. The receiving object does not carry out any other internal action. This service can be used to test whether or not a particular object is still present and responding without causing a state change.

### A-4.17.1 Required Behavior

The NOP service requires the following behaviors:

1. If the object to which the request is delivered supports the service, then a response whose status indicates success is returned. If the object does not support the service, then a response indicating an error was detected is returned.

### A-4.17.2 Request Service Data Field Parameters

NONE

### A-4.17.3 Success Response Service Data Field Parameters

NONE


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.18 Get_Member - Service Code: $18_{Hex}$

Returns member(s) information from within an attribute. See Section A-5 for *Member Service Protocol* details.

### A-4.18.1 Service Requirements

The following list details requirements associated with the Get_Member service:

1. The service causes the class/object to return member(s) at the specified Member ID of an attribute.
2. If an error is detected, then an Error Response is returned. Otherwise, a successful Get_Member response is returned.
3. If *Member ID* is greater than largest existing *Member ID*, gets the one member with the highest Member ID.
4. If the *Member ID* value is zero, returns an *Invalid Member ID* Error Response. To get all members, use a Get_Attribute_Single service.

### A-4.18.2 Request Service Data Field Parameters

**Table A-4.32 Service Data for Get_Member Request**

| Request Field | Required/Optional/Not Present |
|---|---|
| Member Data | Not present |

### A-4.18.3 Success Response Service Data Field Parameters

**Table A-4.33 Service Data for Get_Member Response**

| Response Field | Required/Optional/Not Present |
|---|---|
| Member ID | Required |
| Member Data | Required |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.19 Set_Member - Service Code: $19_{Hex}$

Sets member(s) information in an attribute. See Section A-5 for *Member Service Protocol* details.

### A-4.19.1 Service Requirements

The following list details requirements associated with the Set_Member service:

1. The service causes the class/object to set member(s) at the specified Member ID of an attribute.
2. The request is checked, and if valid, the member(s) are set to the new Member Data.
3. If an error is detected, then an Error Response is returned. Otherwise, a successful Set_Member response is returned.
4. If the Member ID value is greater than that of the last Member ID for the specific attribute, no action occurs and an Invalid Member ID Error Response is returned.
5. If multiple members are specified and fewer members exist at the specified Member ID, no action occurs and an Invalid Member ID Error Response is returned.
6. If the Member ID value is zero, an Invalid Member ID Error Response is returned.

### A-4.19.2 Request Service Data Field Parameters

**Table A-4.34 Service Data for Set_Member Request**

| Request Field | Required/Optional/Not Present |
|---|---|
| Member Data | Required |

### A-4.19.3 Success Response Service Data Field Parameters

**Table A-4.35 Service Data for Set_Member Response**

| Response Field | Required/Optional/Not Present |
|---|---|
| Member ID | Conditional<br>(Required if Member Data is present) |
| Member Data | Optional |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.20 Insert_Member - Service Code: $1A_{Hex}$

Inserts member(s) into an attribute. See Section A-5 for *Member Service Protocol* details.

### A-4.20.1 Service Requirements

The following list details requirements associated with the Insert_Member service:

1. The service causes the class/object to insert member(s) at the specified *Member ID* of an attribute. The Member IDs of members at or following the specified Member ID will change.
2. The request is checked, and if valid, the member(s) are inserted. If member data is not included in the request, the member(s) are set to the class/attribute specific default.
3. If an error is detected, then no action is taken and an Error Response is returned. Otherwise, a successful Insert_Member response is returned.
4. If the number of members specified cannot be added to the attribute, then a "Resource unavailable" error response is returned.
5. If the Member ID value is greater than that of the last *Member ID* for the specific attribute, appends member(s) to the attribute and returns the *Member ID* where inserted.
6. If the *Member ID* value is zero, returns an *Invalid Member ID* Error Response.

### A-4.20.2 Request Service Data Field Parameters

**Table A-4.36 Service Data for Insert_Member Request**

| Request Field | Required/Optional/Not Present |
|---|---|
| Member Data | Optional |

### A-4.20.3 Success Response Service Data Field Parameters

**Table A-4.37 Service Data for Insert_Member Request**

| Response Field | Required/Optional/Not Present |
|---|---|
| Member ID | Required |
| Member Data | Optional |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.21 Remove_Member - Service Code: $1B_{Hex}$

Removes member(s) from an attribute. See Section A-4.22 for *Member Service Protocol* details.

### A-4.21.1 Service Requirements

The following list details requirements associated with the Remove_Member service:

1. The service causes the class/object to remove member(s) at the specified *Member ID* of an attribute. The Member IDs of members following the removed member(s) change.
2. If an error is detected, then no action is taken and an Error Response is returned. Otherwise, a successful Remove_Member response is returned.
3. If *Member ID* is greater than largest existing *Member ID*, the one member with the highest Member ID is removed.
4. If multiple members are specified and fewer members exist at the specified Member ID, the member(s) that do exist are removed.

### A-4.21.2 Request Service Data Field Parameters

**Table A-4.38 Service Data for Remove_Member Request**

| Request Field | Required/Optional/Not Present |
|---|---|
| Member Data | Not present |

### A-4.21.3 Success Response Service Data Field Parameters

**Table A-4.39 Service Data for Remove_Member Response**

| Response Field | Required/Optional/Not Present |
|---|---|
| Member ID | Required |
| Member Data | Required |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-4.22 GroupSync - Service Code: $1C_{Hex}$

Verify each member of a group is synchronized to System Time.

### A-4.22.1 Service Requirements

The following list details requirements associated with the GroupSync service:

1 The structure of the information in the request's Service Data Field adheres to the definition of the GroupSync request for the object or class. Support of this service requires the Object and/or Class to provide a detailed definition of the format of the data sent in the request message.
2 The structure of the information in the response's Service Data Field adheres to the definition of the GroupSync response for the object or class. Support of this service requires the Object and/or Class to provide a detailed definition of the format of the data sent in the response message.
3 The responder will verify that the application is synchronized according to the obect specific requirements. The target return's a 1 in the IsSynchronized attribute of the response Service Data Field if the synchronized status is true and a 0 if the synchronized status is false.

Refer to the GroupSync section of the CIP Sync Overview in the Time Sync Object for a more detailed description of the GroupSync service.

### A-4.22.2 Request Service Data Field Parameters

The following information is specified within the Service Data Field of a GroupSync request.

**Table A-4.40 Service Data for GroupSync Request**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Object specific parameters | Object specific | Object specific |

### A-4.22.3 Response Service Data Field Parameters

The following information is specified within the Service Data Field of a GroupSync response.

**Table A-4.41 Service Data for GroupSync Success Response**

| Name | Data Type | Description of Parameter | Semantics of values |
|---|---|---|---|
| IsSynchronized | BOOL | Indicates if object is synchronized to the PTP Time Master | 1 = Is Synchronized<br>0 = Not Synchronized |
| Object specific parameters | Object specific | Object specific | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# A-5 Member Service Protocols

Attributes of object classes may consist of arrays of basic data types or structures. To manipulate members of an array, the following member services use the protocols defined in this section:

- Get_Member
- Set_Member
- Insert_Member
- Remove_Member

The first member in an array is specified by a *Member ID* value of one (1).

## A-5.1 Member ID/EX Description

Two message formats are defined for member services: basic and extended. Within the extended format, there exist up to 255 protocol options. The message format is selected by the most significant bit of the Member ID. When a subnet does not support Attribute and Member level addressing, the Member ID/EX parameter is sent as a 16 bit (WORD) parameter within the service data. If the subnet does support Attribute and Member level addressing, the Member ID/EX parameter is sent within the Logical Segment and can be either 8, 16, or 32 bits (BYTE, WORD, DWORD). The following table defines the bits contained within the *Member ID/EX* field when sent as a WORD.

**Table A-5.1 Member ID/EX Description (WORD)**

| Bits | | | | | | | |
|---|---|---|---|---|---|---|---|
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Member ID Bits 0-7 | | | | | | | |
| EX | Member ID Bits 8-14 | | | | | | |

The lower fifteen bits (0-14) of the *Member ID/EX* field identify the member to be manipulated. Bit 15 (EX), selects either the basic (EX=0) or extended (EX=1) message format.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-5.2 Member Request Message - Basic

The following table illustrates the basic *Member Request Message* service data format.

**Table A-5.2 Service Data for Basic Format Member Request Messages**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Attribute ID[1] | USINT | Identifies the attribute the member is to be serviced. |
| Member ID/ EX[2] | WORD | The lower 15 bits identifies the Member ID identifies attribute of the new member to be serviced. If the value is zero (0) the service used determines the behavior.<br>Extended Protocol Option EX = 0 (Bit 15) |
| Member Data (conditional) | Member specific | This field is required by some services, see individual service descriptions. |

Table Footnotes

1. This parameter shall be present when the target of the request is on a subnet which does not support explicit message addressing to the attribute level. If the subnet does support attribute level addressing then this parameter shall not be present. In the latter case, the data type of the Attribute ID can be either USINT, UINT, or UDINT.
2. This parameter shall be present when the target of the request is on a subnet, which does not support explicit message addressing to the member or attribute level. If the subnet does support attribute and member level addressing then this parameter shall not be present. In this latter case, the data type of the Member ID/EX can be either BYTE, WORD, or DWORD with the most significant bit being the Extended Protocol Option parameter and the remaining bits being the Member ID.

**Table A-5.3 Service Data for Basic Format Member Response Messages**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Member ID (conditional) | UINT | The Member ID of the member serviced.<br>This field is required by some services, see individual service descriptions. This field is required if the Member Data field is present. |
| Member Data (conditional) | Member specific | This field is required by some services, see individual service descriptions. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## A-5.3 Member Request Message - Extended

If the Extended protocol bit [EX] of the *Member ID/EX* field is set to a one [1], the byte following the *Member ID/EX* defines the remainder of the protocol.

The *Extended Protocol ID* value conforms to the standard reserved ranges for open/vendor specific/ reserved ranges.

**Table A-5.4 Service Data for Extended Format Member Request Messages**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Attribute ID[1] | USINT | Identifies the attribute the member is to be serviced. |
| Member ID / EX[2] | WORD | The lower 15 bits identifies the Member ID identifies attribute of the new member to be serviced. If the value is zero (0) the service used determines the behavior.<br>Extended Protocol Option EX= 1 (Bit 15) |
| Extended Protocol ID | USINT | Selects the extended protocol used for the remainder of message, per Extended Protocol ID table. |
| | Protocol specific | Additional request data is defined by the extended protocol. |

Table Footnotes

1. This parameter shall be present when the target of the request is on a subnet, which does not support explicit message addressing to the attribute level. If the subnet does support attribute level addressing then this parameter shall not be present. In the latter case, the data type of the Attribute ID can be either USINT, UINT, or UDINT.
2. This parameter shall be present when the target of the request is on a subnet, which does not support explicit message addressing to the member or attribute level. If the subnet does support attribute and member level addressing then this parameter shall not be present. In this latter case, the data type of the Member ID/EX can be either BYTE, WORD, or DWORD with the most significant bit being the Extended Protocol Option parameter and the remaining bits being the Member ID.

**Table A-5.5 Service Data for Extended Format Member Response Messages**

| Name | Data Type | Description of Parameter |
|---|---|---|
| | Protocol specific | Response data is defined by the extended protocol option. |

The following table contains a list of the currently defined values for the Extended Protocol ID.

**Table A-5.6 Extended Protocol ID**

| Value | Description |
|---|---|
| 0 | Reserved for future CIP use |
| 1 | Multiple Sequential Members |
| 2 | International String Selection |
| 3-63hex | Reserved for future CIP use |
| 64hex-C7hex | Vendor Specific |
| C8hex-FFhex | Reserved for future CIP use |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-5.3.1 Multiple Sequential Members - Extended Protocol

When the *Extended Protocol ID* is set to *Multiple Sequential Members* [1], the following protocol is used:

**Table A-5.7 Service Data for Multiple Sequential Member Request Messages**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Attribute ID[1] | USINT | Identifies the attribute the member is to be serviced. |
| Member ID / EX[2] | WORD | The lower 15 bits identifies the Member ID identifies attribute of the new member to be serviced. If the value is zero (0) the service used determines the behavior.<br>Extended Protocol Option EX= 1 (Bit 15) |
| Extended Protocol ID | USINT [01] | Selects the Multiple Sequential Members protocol option |
| Number of Members | UINT | The number of members to be serviced. |
| Member Data (conditional) | Member specific | Multiple sequential Member Data<br>This field is required by some services, see individual service descriptions. |

Table Footnotes

1. This parameter shall be present when the target of the request is on a subnet, which does not support explicit message addressing to the attribute level. If the subnet does support attribute level addressing then this parameter shall not be present. In this latter case, the data type of the Attribute ID can be either USINT, UINT, or UDINT.
2. This parameter shall be present when the target of the request is on a subnet, which does not support explicit message addressing to the member or attribute level. If the subnet does support attribute and member level addressing then this parameter shall not be present. In the latter case, the data type of the Member ID/EX can be either BYTE, WORD, or DWORD with the most significant bit being the Extended Protocol Option parameter and the remaining bits being the Member ID.

**Table A-5.8 Service Data for Member Response Messages (Extended Protocol)**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Number of Members | UINT | The number of members serviced. This field is required. |
| Member ID (conditional) | UINT | The Member ID of the first member serviced.<br>This field is required by some services, see individual service descriptions. This field is required if the Member Data field is present. |
| Member Data (conditional) | Member specific | Multiple sequential Member Data<br>This field is required by some services, see individual service descriptions. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-5.3.2 International String Selection– Extended Protocol

Returns a single international character string from an attribute, which has a data type of STRINGI. This service can request a specific language or default to the current 'active' language as specified by the Identity Object.

Service Requirements

The International String Selection protocol requires the following behavior:

1. This member protocol causes the class/object to return an international string from an attribute of data type STRINGI.

2. This member protocol is only valid with the Get_Member service.

3. If the object *or* attribute to which the request is delivered does not support the service/protocol, then a Service Not Supported error is returned.

4. If the attribute does not contain the requested language, then an Invalid Parameter error is returned.

5. If the Member ID parameter is zero, then the device shall return the active (default) language. This default language is set by the device and may optionally be changed through the Active Language attribute (Attribute ID 11) within the Identity object.

6. A device shall support the Supported Language List attribute (Attribute ID 12) of the Identity object if this member protocol is supported.

7. If the requested Member ID is valid but the string for that language is not currently available then an error shall be returned indicating Resource Unavailable (error code 0x02) and no additional error code. The string for the active language (as indicated by the Active Language attribute of the Identity Object) shall always be available.

**Table A-5.9 Service Data for International String Selection Request Messages**

| Name | Data Type | Description of Parameter |
|---|---|---|
| Attribute ID | USINT | Identifies the attribute containing the international string. |
| Member ID / EX | WORD | The lower 15 bits identifies the language to retrieve. This value is the member ID (array index, starting at one) of the language within the Supported Language List attribute (Attribute ID 12) of the Identity object.<br><br>Extended Protocol Option EX = 1 (Bit 15) |
| Extended Protocol ID | USINT | Selects the International String Selection protocol option (value = 2). |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table A-5.10 Service Data for International String Selection Response Messages**

| Name | Data Type | Description of Parameter |
|---|---|---|
| International String | STRUCT of | The character string in the requested language. This parameter follows the definition of a single international character string within the STRINGI data type (Appendix C-4 Abstract Syntax Specification). |
| | USINT | The language1 field from the STRINGI data type. |
| | USINT | The language2 field from the STRINGI data type. |
| | USINT | The language3 field from the STRINGI data type. |
| | USINT | The structure of the character string, limited to the values 0xD0, 0xD5, 0xD9 and 0xDA. |
| | UINT | Character set of the international string. |
| | OCTET_STRING | International string |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# A-6 CIP Encoding Examples

The example object definitions presented in the following section are used as the basis for providing CIP Common Service encoding examples.

## A-6.1 Example Object Class Definitions

Table A-6.1 lists the class codes of the example Object Classes.

**Table A-6.1 Class ID Codes For Example Classes**

| Class Name | Class ID Code |
|---|---|
| Class A | $70_{hex}$ |
| Class B | $71_{hex}$ |
| Class C | $72_{hex}$ |
| Class D | $73_{hex}$ |

The tables below illustrate the attributes associated with Object Instances within these classes.

**Table A-6.2 Class A Object Instance Attributes**

| Attribute Name | Attribute ID | Format | Access Method | Attribute Description |
|---|---|---|---|---|
| attribute_1 | 1 | UINT | Gettable/ Settable | Allows values in range of 1 - 9 |
| attribute_2 | 2 | SINT | Gettable | No limit to value |
| attribute_3 | 3 | DINT | Gettable/ Settable | No limit to value |

**Table A-6.3 Class B Object Instance Attributes**

| Attribute Name | Attribute ID | Format | Access Method | Attribute Description |
|---|---|---|---|---|
| attribute_1 | 1 | DINT | Gettable/ Settable | No limit to value |
| attribute_2 | 2 | SINT | Gettable | No limit to value |
| attribute_3 | 3 | UINT | Gettable/ Settable | Value limited to range of 10 - 20 |
| attribute_4 | 4 | UINT | Gettable/ Settable | Value limited to range of 0 - 9 |

**Table A-6.4 Class C Object Instance Attributes**

| Attribute Name | Attribute ID | Format | Access Method | Attribute Description |
|---|---|---|---|---|
| attribute_1 | 1 | UINT | Gettable | No limit to value |
| attribute_2 | 2 | UINT | Gettable | No limit to value |
| attribute_3 | 3 | UINT | Gettable/ Settable | Allows values in range of 1 - 5 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table A-6.5 Class D Object Instance Attributes**

| Attribute Name | Attribute ID | Format | Access Method | Attribute Description |
|---|---|---|---|---|
| attribute_1 | 1 | USINT | Gettable/ Settable | No limit to value |
| attribute_2 | 2 | UINT | Gettable/ Settable | No limit to value |

## A-6.2 Encoding Examples

Examples showing how the CIP Common Services are encoded are provided on the following pages. Note that the examples assume that an Explicit Messaging Connection has already been established and, as such, the Connection IDs associated with the Explicit Messaging Connections have already been agreed upon.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.1 Get_Attributes_All

The Client wants to read all of the attributes from Instance #1 within Class D with one request. Assume the Explicit Messaging Connection was established across Group 3 and the Message Body Format was defined by the Server as DeviceNet (8/8).

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.2 Set_Attributes_All

The Client wants to modify all of the attributes from Instance #2 within Class D with one request. Assume the Explicit Messaging Connection was established across Group 3 and the Message Body Format was defined by the Server as DeviceNet (8/8).

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.3 Reset

The Client wants to Reset Instance #4 within Class B. Assume the Explicit Messaging Connection was established across Group 3 and the Message Body Format was defined by the Server as DeviceNet (16/16).

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.4 Start

The Client wants to Start Instance #5 within Class C. Assume the Explicit Messaging Connection was established across Group 1 and the Message Body Format was defined by the Server as DeviceNet (8/16).

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.5 Stop

The Client wants to Stop Instance #9 within Class B. Assume the Explicit Messaging Connection was established across Group 3 and the Message Body Format was defined by the Server as DeviceNet (8/8).

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.6 Create

The Client wants to Create an instance within Class A. Assume the Explicit Messaging Connection was established across Group 2 and the Message Body Format was defined by the Server as DeviceNet (8/8)

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.7 Delete

The Client wants to Delete Instance #3 within Class B. Assume the Explicit Messaging Connection was established across Group 3 and the Message Body Format was defined by the Server as DeviceNet (8/16).

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.8 Apply_Attributes

The Client wants to send an Apply_Attributes to Instance #2 within Class C. Assume the Explicit Messaging Connection was established across Group 2 and the Message Body Format was defined by the Server as DeviceNet (16/16).

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.9 Get_Attribute_Single

The Client wants to read Attribute #1 within Instance #2 of Class A. Assume the Explicit Messaging Connection was established across Group 1 and the Message Body Format was defined by the Server as DeviceNet (8/8).

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.10 Set_Attribute_Single

The Client wants to modify Attribute #1 within Instance #2 of Class A. Assume the Explicit Messaging Connection was established across Group 2 and the Message Body Format was defined by the Server as DeviceNet (8/16).

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.11 Find_Next_Object_Instance

The Client wants to read the list of Instance IDs associated with existing Objects within Class A. Assume the Explicit Messaging Connection was established across Group 2 and the Message Body Format was defined by the Server as DeviceNet (8/8). Assume also that the Instance IDs $01_{hex}$ and $15_{hex}$ exist.

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.12 Restore

The Client wants to send a Restore Request to Instance #3 within Class B. Assume the Explicit Messaging Connection was established across Group 3 and the Message Body Format was defined by the Server as DeviceNet (8/8).

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.13 Save

The Client wants to send a Save Request to Instance #5 within Class C. Assume the Explicit Messaging Connection was established across Group 1 and the Message Body Format was defined by the Server as DeviceNet (16/16).

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.14 No Operation

The Client wants to send a No Operation (NOP) to Instance #2 within Class C. Assume the Explicit Messaging Connection was established across Group 2 and the Message Body Format was defined by the Server as DeviceNet (16/16).

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.15 Error Response

The Client wants to Reset Instance #1 within Class D. Assume the Explicit Messaging Connection was established across Group 1 and the Message Body Format was defined by the Server as DeviceNet (8/8). Assume also that Objects within Class D do not support the Reset service.

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.16 Get_Member

The Client wants to read Member #4 of Attribute #3 within Instance #2 of Class D. Assume the Explicit Messaging Connection was established across Group 1 and the Message Body Format was defined by the Server as DeviceNet (8/8).

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.17 Set_Member

The Client wants to modify Member #5 of Attribute #3 within Instance #1 of Class D. Assume the Explicit Messaging Connection was established across Group 2 and the Message Body Format was defined by the Server as DeviceNet (8/8).

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.18 Insert_Member

The Client wants to Add Member #14 of Attribute #3 within Instance #3 of Class D. Assume the Explicit Messaging Connection was established across Group 1 and the Message Body Format was defined by the Server as DeviceNet (8/8).

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### A-6.2.19 Remove_Member

The Client wants to remove Member #9 of Attribute #3 within Instance #1 of Class D. Assume the Explicit Messaging Connection was established across Group 2 and the Message Body Format was defined by the Server as DeviceNet (8/8).

SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This page is intentionally left blank


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Volume 1: Common Industrial Protocol (CIP™) Specification

# Appendix B: Status Codes


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Contents

B-1 General Status Codes ..... 3


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# B-1 General Status Codes

The following table lists the Status Codes that may be present in the General Status Code field of an Error Response message. Note that the Extended Code Field is available for use in further describing any General Status Code. Extended Status Codes are unique to each General Status Code within each object. Each object shall manage the extended status values and value ranges (including vendor specific). All extended status values are reserved unless otherwise indicated within the object definition.

These codes may also be used for other purposes such as event logging (e.g., via the Event Log object) or in device heartbeat messages.

**Table B-1.1 CIP General Status Codes**

| General Status Code (in hex) | Status Name | Description of Status |
|---|---|---|
| 00 | Success | Service was successfully performed by the object specified. |
| 01 | Connection failure | A connection related service failed along the connection path. |
| 02 | Resource unavailable | Resources needed for the object to perform the requested service were unavailable |
| 03 | Invalid parameter value | See Status Code 0x20, which is the preferred value to use for this condition. |
| 04 | Path segment error | The path segment identifier or the segment syntax was not understood by the processing node. Path processing shall stop when a path segment error is encountered. |
| 05 | Path destination unknown | The path is referencing an object class, instance or structure element that is not known or is not contained in the processing node. Path processing shall stop when a path destination unknown error is encountered. |
| 06 | Partial transfer | Only part of the expected data was transferred. |
| 07 | Connection lost | The messaging connection was lost. |
| 08 | Service not supported | The requested service was not implemented or was not defined for this Object Class/Instance. |
| 09 | Invalid attribute value | Invalid attribute data detected |
| 0A | Attribute list error | An attribute in the Get_Attribute_List or Set_Attribute_List response has a non-zero status. |
| 0B | Already in requested mode/state | The object is already in the mode/state being requested by the service |
| 0C | Object state conflict | The object cannot perform the requested service in its current mode/state |
| 0D | Object already exists | The requested instance of object to be created already exists. |
| 0E | Attribute not settable | A request to modify a non-modifiable attribute was received. |
| 0F | Privilege violation | A permission/privilege check failed |
| 10 | Device state conflict | The device's current mode/state prohibits the execution of the requested service. |
| 11 | Reply data too large | The data to be transmitted in the response buffer is larger than the allocated response buffer |
| 12 | Fragmentation of a primitive value | The service specified an operation that is going to fragment a primitive data value, i.e. half a REAL data type. |
| 13 | Not enough data | The service did not supply enough data to perform the specified operation. |
| 14 | Attribute not supported | The attribute specified in the request is not supported |
| 15 | Too much data | The service supplied more data than was expected |
| 16 | Object does not exist | The object specified does not exist in the device. |
| 17 | Service fragmentation sequence not in progress | The fragmentation sequence for this service is not currently active for this data. |
| 18 | No stored attribute data | The attribute data of this object was not saved prior to the requested service. |
| 19 | Store operation failure | The attribute data of this object was not saved due to a failure during the attempt. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| General Status Code (in hex) | Status Name | Description of Status |
|---|---|---|
| 1A | Routing failure, request packet too large | The service request packet was too large for transmission on a network in the path to the destination. The routing device was forced to abort the service. |
| 1B | Routing failure, response packet too large | The service response packet was too large for transmission on a network in the path from the destination. The routing device was forced to abort the service. |
| 1C | Missing attribute list entry data | The service did not supply an attribute in a list of attributes that was needed by the service to perform the requested behavior. |
| 1D | Invalid attribute value list | The service is returning the list of attributes supplied with status information for those attributes that were invalid. |
| 1E | Embedded service error | An embedded service resulted in an error. |
| 1F | Vendor specific error | A vendor specific error has been encountered. The Additional Code Field of the Error Response defines the particular error encountered. Use of this General Error Code should only be performed when none of the Error Codes presented in this table or within an Object Class definition accurately reflect the error. |
| 20 | Invalid parameter | A parameter associated with the request was invalid. This code is used when a parameter does not meet the requirements of this specification and/or the requirements defined in an Application Object Specification. |
| 21 | Write-once value or medium already written | An attempt was made to write to a write-once medium (e.g. WORM drive, PROM) that has already been written, or to modify a value that cannot be changed once established. |
| 22 | Invalid Reply Received | An invalid reply is received (e.g. reply service code does not match the request service code, or reply message is shorter than the minimum expected reply size). This status code can serve for other causes of invalid replies. |
| 23 | Buffer Overflow | The message received is larger than the receiving buffer can handle. The entire message was discarded. |
| 24 | Message Format Error | The format of the received message is not supported by the server. |
| 25 | Key Failure in path | The Key Segment that was included as the first segment in the path does not match the destination module. The object specific status shall indicate which part of the key check failed. |
| 26 | Path Size Invalid | The size of the path which was sent with the Service Request is either not large enough to allow the Request to be routed to an object or too much routing data was included. |
| 27 | Unexpected attribute in list | An attempt was made to set an attribute that is not able to be set at this time. |
| 28 | Invalid Member ID | The Member ID specified in the request does not exist in the specified Class/Instance/Attribute |
| 29 | Member not settable | A request to modify a non-modifiable member was received |
| 2A | Group 2 only server general failure | This error code may only be reported by DeviceNet Group 2 Only servers with 4K or less code space and only in place of Service not supported, Attribute not supported and Attribute not settable. |
| 2B | Unknown Modbus Error | A CIP to Modbus translator received an unknown Modbus Exception Code. |
| 2C | Attribute not gettable | A request to read a non-readable attribute was received |
| 2D | Instance Not Deletable | The requested object instance cannot be deleted |
| 2E | Service Not Supported for Specified Path [1] | The object supports the service, but not for the designated application path (e.g. attribute).<br>NOTE: Not to be used for any set service (use General Status Code 0x0E or 0x29 instead) |
| 2F - CF | | Reserved by CIP for future extensions |
| D0 - FF | Reserved for Object Class specific errors | This range of error codes is to be used to indicate Object Class specific errors. Use of this range should only be performed when none of the Error Codes presented in this table accurately reflect the error that was encountered. |

Table Footnotes

1. Use of this general status code shall be enforced effective January 1, 2014.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Volume 1: Common Industrial Protocol (CIP™) Specification

# Appendix C: Data Management


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Contents

C-1 Abstract Syntax Encoding Coding for Segment Types ..... 3
C-1.1 CIP Segments ..... 3
C-1.2 Usage of Segments for Paths ..... 3
C-1.3 Path Format ..... 4
C-1.4 Segment Type ..... 5
C-1.4.1 Port Segment ..... 6
C-1.4.2 Logical Segment ..... 8
C-1.4.3 Network Segment ..... 12
C-1.4.4 Symbolic Segment ..... 15
C-1.4.5 Data Segment ..... 16
C-1.5 Segment Definition Hierarchy ..... 17
C-1.6 Encoded Path Compression Rules ..... 19
C-2 Data Type Specification ..... 20
C-2.1 Data Type Values ..... 20
C-2.1.1 Elementary Data Types ..... 21
C-2.1.2 Character String Data Types ..... 22
C-2.1.3 Structured Bit String Types ..... 24
C-2.1.4 Derived Data Types ..... 24
C-2.1.5 Defined Constructed Data Types ..... 24
C-3 IEC 1131-3 Compliance ..... 28
C-3.1 Compliance Statement ..... 28
C-3.2 Implementation-Dependent Parameters ..... 32
C-3.3 Language Extensions ..... 32
C-4 Abstract Syntax Specification ..... 33
C-4.1 CIP Data Specification ..... 33
C-4.2 Data Type Specification/Dictionaries ..... 35
C-5 CIP Application Transfer Syntax: Compact Encoding ..... 38
C-5.1 Compact Encoding Constraints ..... 38
C-5.2 Encoding Rules/Examples ..... 38
C-5.2.1 BOOL Encoding ..... 38
C-5.2.2 SignedInteger Encoding ..... 39
C-5.2.3 UnsignedInteger Encoding ..... 39
C-5.2.4 FixedLengthReal Encoding ..... 40
C-5.2.5 Time Encodings ..... 40
C-5.2.6 String Encodings ..... 40
C-5.2.7 FixedLengthBitString Encoding ..... 42
C-5.2.8 Array Encoding ..... 43
C-5.2.9 Structure Encoding ..... 44
C-5.2.10 Examples of How More Complex Data Formats are Encoded ..... 44
C-6 Data Type Reporting ..... 46
C-6.1 Elementary Data Type Reporting ..... 47
C-6.2 Constructed Data Type Reporting ..... 48
C-6.2.1 Structure Type Definition ..... 48
C-6.2.2 Array Type Definition ..... 51
C-6.2.3 Defined Constructed Data Types ..... 53
C-7 CRC Generation Algorithms ..... 55


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# C-1 Abstract Syntax Encoding Coding for Segment Types

## C-1.1 CIP Segments

This section specifies the encoding for CIP segments. A CIP segment is a stream of encoded items used to reference, describe, and/or configure a specific CIP entity. Segments are usually grouped together in order to provide complete information.

The encoding used for CIP segments has the following relationship with the encoding used to report CIP data types, thus allowing both to be intermixed:

**Table C-1.1 Segment Encoding/Data Reporting Relationships**

| Encoding of 1st Byte | Description |
|---|---|
| 00-9F hex | Encoding for Segments |
| A0-DF hex | Encoding for Data Type Reporting |
| E0-FF hex | Reserved for future use |

A segment contains information indicating what segment type is specified, the format of the segment data, and the actual segment data. Encoding for the following segment types is described in section C-1.4:

- Port segment – used for routing from one subnet to another
- Logical segment - logical reference information (such as class/instance/attribute IDs)
- Network segment - specifies network parameters needed to transmit on some networks
- Symbolic segment - symbolic name
- Data segment - embedded data (such as configuration data)

Rules for how the segments are ordered and the interpretation of their meaning are described in section C-1.5.

## C-1.2 Usage of Segments for Paths

A common use of CIP segments is to specify relationships among different objects. A value used to specify such a relationship is referred to as a path. A path attribute consists of multiple segments, and typically references the class, instance, and attribute of another object. A path has data type EPATH (Packed or Padded).

Some examples of how paths are used include:

- In Connection and Connection Manager Objects, paths indicate the object(s) to/from which I/O data is moved.
- In Assembly Objects, paths indicate the attributes in other objects which are used to form the assembled I/O data.
- In Parameter Objects, paths indicate the actual attribute in another object which is being described by the Parameter Object.

An example of a path attribute is shown in Figure C-1.1. For an example encoding for such a path, refer to section C-1.4.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## C-1.3 Path Format

**Figure C-1.1 Example Path Attribute**

A path (data type EPATH) can be represented in two different formats:

Padded Path (indicated as data type Padded EPATH)

Packed Path (indicated as data type Packed EPATH)

Each segment of a Padded Path shall be 16-bit word aligned. If a pad byte is required to achieve the alignment, the segment shall specify the location of the pad byte. A Packed Path shall not contain any pad bytes. When a component is defined as data type EPATH, it shall indicate the format (Packed or Padded).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## C-1.4 Segment Type

Each segment of an encoding contains a segment type/format byte, which indicates how the segment is to be interpreted. The segment type/format information is contained in the first byte of the segment and, as such, the first byte of any encoding contains a segment type/format byte.

**Figure C-1.2 Segment Type/Format byte Encoding**

| Segment Type | | | Segment Format | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | | | | | |

The *Segment Type* bits are described below:

| Segment Type | | | |
|---|---|---|---|
| 7 | 6 | 5 | |
| 0 | 0 | 0 | Port Segment |
| 0 | 0 | 1 | Logical Segment |
| 0 | 1 | 0 | Network Segment |
| 0 | 1 | 1 | Symbolic Segment |
| 1 | 0 | 0 | Data Segment |
| 1 | 0 | 1 | Data Type (constructed, see section C-6.2) |
| 1 | 1 | 0 | Data Type (elementary, see section C-6.1) |
| 1 | 1 | 1 | Reserved for future use |

The meaning of the *Segment Format* bits is based on the specified *Segment Type*.

The Port Segment, Logical Segment, Network Segment, Symbolic Segment, and Data Segment are described in sections C-1.4.1 to C-1.4.5.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### C-1.4.1 Port Segment

The port segment shall indicate

Communication port through which to leave the node;

Link address of the next device in the routing path.

The Port Segment is formatted as defined in the figure below. The bit representation is from high bit to low bit, left to right. The byte representation is from low byte to high byte, top to bottom and left to right.

**Figure C-1.3 Port Segment Encoding**

Bit 4, the **Extended Link Address Size** bit, shall be set to 0 if the link address is one byte. Bit 4 shall be set to 1 if the link address is larger than one byte. If the link address is larger than one byte, its size in bytes shall be in the second byte of the Port Segment.

To support the Port Object instance attribute 7 – Port Number and Node Address (see section 3-7), a port segment with no link address can be specified by setting the **Extended Link Address Size** bit (Bit 4 set to 1) and setting the **Optional Link Address Size** byte to 0.

Bits 0 – 3, the **Port Identifier**, shall indicate through which port to leave the node. The Port Identifier shall specify a *Port Number* or an escape to an extended port identifier when the module can support more than 14 ports. The list below defines rules pertinent to Port Identification/Port Number and Link Address values:

Port Number 0 shall be reserved.

Port Number 1 shall only be used to represent a back-plane port.

If the Port Identifier is 15, then a 16-bit field, called the Extended Port Identifier, shall be the next part of the Port Segment (following the optional Link Address Size if present); otherwise, the value of the Port Identifier shall be the Port Number.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The Port Number shall be followed by a **Link Address** whose format depends on the type of network to which the Port Identifier refers. If the Link Address is greater than one byte it shall be padded so that the length of the entire port segment is an even number of bytes. The pad byte shall be set to zero and shall not be included in the Link Address Size. For example, with respect to the specification of a DeviceNet Port, the Link Address shall indicate the target DeviceNet MAC ID and be specified in a single byte whose value is 0 – 63 (the valid range of DeviceNet MAC Ids).

The Extended Port Identifier format of the Port Segment shall only be used when there are more than 14 network ports possible on the connecting device. The Port Segment shall always be represented in the smallest Port Segment format possible with respect to the optional fields.

**Examples** of Port Segments are presented below.

**Table C-1.2 Port Segment Examples**

| Segment contents | Notes |
|---|---|
| [02][06] | Segment Type = Port Segment. Port Number = 2, Link Address = 6 |
| [0F][12][00][01] | Segment Type = Port Segment. Port Identifier is 15 indicating the Port Number is specified in the next 16 bit field [12][00] (18 decimal). Link Address = 1. |
| [15][0F][31][33][30][2E]<br>[31][35][31][2E][31][33]<br>[37][2E][31][30][35][00] | Segment Type = Port Segment. Multi-Byte address for TCP Port 5, Link Address 130.151.137.105 (IP Address). The address is defined as a character array, length of 15 bytes. The last byte in the segment is a pad byte.<br>Note: For a detailed description of encoding IP Addresses in Port Segments, see Volume 2, Section 3-3.7 Connection Path. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### C-1.4.2 Logical Segment

The logical segment selects a particular object address within a device (for example, Object Class, Object Instance, and Object Attribute). When the logical segment is included within a Packed Path, the Logical Value shall be appended to the segment type byte with no pad in between.

**Figure C-1.4 Logical Segment Encoding**

| | | | | | | |
|---|---|---|---|---|---|---|
| Class ID | 0 | 0 | 0 | 0 | 0 | 8-bit logical value |
| Instance ID | 0 | 0 | 1 | 0 | 1 | 16-bit logical value |
| Member ID | 0 | 1 | 0 | 1 | 0 | 32-bit logical value |
| Connection Point | 0 | 1 | 1 | 1 | 1 | Reserved for future use |
| Attribute ID | 1 | 0 | 0 | | | |
| Special [1] | 1 | 0 | 1 | | | |
| Service ID [1] | 1 | 1 | 0 | | | |
| Extended Logical | 1 | 1 | 1 | | | |

Figure Footnotes
1 The Special and Service ID Logical Types do not use the logical addressing definition for the Logical Format.
2 Required when Logical Type is Extended Logical (1 1 1), not present otherwise. The Extended Logical Type values are defined in Table C-1.3.
3 Depends on value of Logical Format.

The Extended Logical Types are enumerated in the table below.

**Table C-1.3 Extended Logical Type**

| Value | Description |
|---|---|
| 0 | Reserved |
| 1 | Array Index |
| 2 | Indirect Array Index |
| 3 | Bit Index |
| 4 | Indirect Bit Index |
| 5 | Structure Member Number |
| 6 | Structure Member Handle |
| 7-255 | Reserved |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The *8-bit and 16-bit logical address* formats are allowed for use with all Logical Types. The *32-bit logical address* format is only allowed for the logical Instance ID, Connection Point, Array Index, Bit Index, Structure Member Number and Structure Member Handle types. It is not allowed for any other Logical Type (reserved for future use).

When an extended logical segment is included within a Padded Path and the Logical Format is 8 bit, a pad byte shall be added after the Logical Value (the 16-bit and 32-bit formats are identical to the Packed Path) and shall be set to 0. For all other logical segments, when included within a Padded Path, the 16-bit and 32-bit logical formats shall have a pad inserted between the segment type byte and the Logical Value (the 8-bit format is identical to the Packed Path). The pad byte shall be set to zero.

When an Extended Logic Segment is specified the Extended Logical Segment Value byte shall be included between the Segment Type byte and the Logical Value.

An Array Index Extended Logical Segment is used to specify the 0-based index into the preceding item in the EPath which must be an array.

An Indirect Array Index Extended Logical Segment is used to specify the start of a nested application path, which resolves to a 0-based index into the preceding item in the EPath which must be an array. The Indirect Array Index Logical Segment Logical Value is the size of the nested application path, in 16-bit words.

A Bit Index Extended Logical Segment is used to specify the 0-based bit number into the preceding item in the EPath (there is no restriction on type of the preceding item).

An Indirect Bit Index Extended Logical Segment is used to specify the start of a nested application path, which resolves to a 0-based bit number into the preceding item in the EPath (there is no restriction on type of the preceding item). The Indirect Bit Index Logical Segment Logical Value is the size of the nested application path, in 16-bit words.

A Structure Member Number Extended Logical Segment is used to specify the 1-based member number into the preceding item in the EPath which must be a structure. The member number specifies the structure member, where the structure members are numbered starting with 1.

A Structure Member Handle Extended Logical Segment is used to specify the member handle into the preceding item in the EPath which must be a structure. The handle is a value that uniquely identifies a member and is provided by the target device when reporting constructed data type information using the Formal Encoding with Handles specified in section C-6.2.1.

The *Connection Point* Logical Type provides additional addressing capabilities beyond the standard Class ID/Instance ID/Attribute ID/Member ID Object Address. *Object Classes shall define when and how this addressing component is utilized*.

The *Service ID* Logical Type has the following definition for the Logical Format:

0 0 8-Bit Service ID Segment (0x38)
0 1 Reserved for future use (0x39)
1 0 Reserved for future use (0x3A)
1 1 Reserved for future use (0x3B)

The *Special* Logical Type has the following definition for the Logical Format:

0 0 Electronic Key Segment (0x34)
0 1 Reserved for future use (0x35)
1 0 Reserved for future use (0x36)
1 1 Reserved for future use (0x37)


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The *Electronic Key* segment shall be used to verify/identify a device. Possible uses include verification during connection establishment and identification within an EDS file. This segment has the format as shown in the table below.

**Table C-1.4 Electronic Key Segment Format**

| Field Name | Data Type | Semantics |
|---|---|---|
| Segment Type | USINT | A value of 0x34 indicates a Logical Electronic Key segment |
| Key Format | USINT | 0 – 3 = Reserved<br>4 = see Key Format Table<br>5 – 255 = Reserved |
| Key Data | Array of octet | Depends on key format used |

**Table C-1.5 Key Format Table**

| Format Value | Field Name | Data Type | Semantics |
|---|---|---|---|
| 4 | Vendor ID | UINT | Vendor ID |
| | Device Type | UINT | Device Type |
| | Product Code | UINT | Product Code |
| | Major Revision / Compatibility | BYTE | Bits 0 – 6 = Major Revision<br>Bit 7 = Compatibility<br>0 (clear) – any non-zero Vendor ID, Device Type, Product Code, Major Revision, and Minor Revision shall match.<br>1 (set) – any key may be accepted which a device can emulate [1] |
| | Minor Revision | USINT | Minor Revision |

Table Footnotes

1 When the Compatibility bit is set (1) the Vendor ID, Device Type, Product Code and Revision are used to identify the device the recipient is being requested to emulate; therefore 0 is an invalid value for the Vendor ID, Device Type, Product Code, Major Revision and Minor Revision fields.

If the Vendor ID, Device Type, Product Code, Major Revision and/or Minor Revision is 0 or the device cannot emulate the requested device, the request shall be rejected. If the compatible key was included in a request directed at the Message Router object, then one of the errors in Table 5A-3.10 Electronic Key Segment Error Response Values shall be returned. If the request was directed at the Connection Manager object, then one of the key-specific errors (general status 0x01 and extended status 0x0114 through 0x0116) in Table 3-5.34 Connection Manager Service Error Codes shall be returned.

EXCEPTION: Device Type 0 is allowed if the recipient can emulate a device that reports the deprecated Generic Device Type Number.

An encoding that specifies Class #5, Instance #2, and Attribute ID #1 is illustrated below (Note that the packed and padded representations are the same):


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure C-1.5 <u>Packed</u> EPATH with 8 bit Class**

The same example is presented below, except that the Class information is specified in a 16 bit field. Figure C-1.6 shows a packed representation and Figure C-1.7 shows padded.

**Figure C-1.6 <u>Packed</u> EPATH with 16 bit Class**

**Figure C-1.7 <u>Padded</u> EPATH with 16 bit Class**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 21 | 00 | 05 | 00 | 24 | 02 | 30 | 01 |

▲ Pad inserted here for padded representation


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### C-1.4.3 Network Segment

The network segment is used to specify network parameters that may be required by a node to transmit a message across a network.

**Figure C-1.8 Network Segment Encoding**

| Network Segment Type | Segment Sub-type | Network Segment Data | |
|---|---|---|---|
| 0 1 0 | 0 | Data (One Byte) | |
| 0 1 0 | 1 | Number of Data Words (One byte) | Data (Variable length) |

↓

| | |
|---|---|
| 0 0 0 0 0 | Reserved for future use |
| 0 0 0 0 1 | Schedule Segment |
| 0 0 0 1 0 | Fixed Tag Segment |
| 0 0 0 1 1 | Production Inhibit Time in Milliseconds |
| 0 0 1 0 0 | Reserved for future use |
| thru | |
| 0 1 1 1 1 | Reserved for future use |
| 1 0 0 0 0 | Safety Segment |
| 1 0 0 0 1 | Production Inhibit Time in Microseconds |
| 1 0 0 1 0 | Reserved for future use |
| thru | |
| 1 1 1 1 0 | Reserved for future use |
| 1 1 1 1 1 | Extended Network Segment |

#### C-1.4.3.1 Schedule Network Segment

Appendix C-2.1 of Volume 4 (ControlNet Adaptation of CIP) defines the Schedule network segment

#### C-1.4.3.2 Fixed Tag Network Segment

Appendix C-2.2 of Volume 4 (ControlNet Adaptation of CIP) defines the Fixed Tag network segment.

#### C-1.4.3.3 Production Inhibit Time Network Segments

The production inhibit time network segments shall specify the minimum time between successive transmissions of new data for the specified connection. The Production Inhibit Time Network Segments only apply to Change of State or Application triggered connections. A value of zero shall indicate no production inhibit time. When a production inhibit segment is not provided during establishment of a connection a default value of one-fourth the RPI shall be used.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The production inhibit time shall be less than or equal to the RPI. If the production inhibit time is greater than the RPI, the Forward_Open response shall be returned with one of the following errors:

- Production Inhibit Time is greater than the RPI. General Status 0x01, Extended Status 0x011B (recommended)
- RPI Not Supported. General Status 0x01, Extended Status 0x0111 (deprecated)

The production inhibit time network segment only applies to the target device.

Target devices may support neither of the Production Inhibit Time network segments. If the device supports the Production Inhibit Time in Microseconds Network Segment it shall also support the Production Inhibit Time in Milliseconds Network Segment.

#### C-1.4.3.3.1 Production Inhibit Time in Milliseconds Network Segment

The Production Inhibit Time in Milliseconds Network Segment specifies a time in milliseconds.

The time specified in the Production Inhibit Time in Milliseconds Network Segment corresponds to the production inhibit definition in the Chapter 3, Connection Object production_inhibit_time attribute (17) definition.

For example, if a production inhibit time of 10 milliseconds is specified, new data shall be sent no sooner than 10 milliseconds after the previous new data production.

The network segment data field shall be a single USINT (8 bits) indicating the production inhibit time in milliseconds. The production inhibit time range is 0 to 255 milliseconds.

#### C-1.4.3.3.2 Production Inhibit Time in Microseconds Network Segment

The Production Inhibit Time in Microseconds Network Segment specifies a time in microseconds.

The time specified in the Production Inhibit Time in Microseconds Network Segment is used in the manner defined in the production inhibit definition in the Chapter 3, Connection Object production_inhibit_time attribute (17) definition except that the production inhibit time is in microseconds instead of milliseconds and is a UDINT instead of a USINT.

For example, if a production inhibit time of 250 microseconds is specified, new data shall be sent no sooner than 250 microseconds after the previous new data production.

The Network Segment Data field shall consist of:

- a USINT indicating the Number of Data Words field. The value shall be 2.
- a UDINT (32 bits) indicating the production inhibit time in microseconds, the production inhibit time range is 0 to 4294967295 microseconds.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### C-1.4.3.4 Extended Network Segment

The Extended Network Segment allows for the definition of additional Network Segment Subtypes. The first word of data is the Extended Network Segment Subtype.

**Figure C-1.9 Extended Network Segment Format**

| Network Segment Type | | | Segment Sub-type | | | | | Network Segment Data | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | Number of Data Words (One byte) | Extended Network Segment Subtype (1 word) | Data (Variable length) |

Each extended subtype defines the data that follows. The extended subtypes are enumerated in the table below.

**Table C-1.6 Extended Subtype Value Definition**

| Extended Subtype Value | Extended Subtype Definition |
|---|---|
| 0 – 32767 | Open, Reserved for future use |
| 32768 – 65535 | Vendor Specific |

### C-1.4.3.5 Safety Network Segment

The Safety network segment is defined in the CIP Safety Specification (Volume 5, Appendix C)


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### C-1.4.4 Symbolic Segment

The symbolic segment contains an International String symbol which must be interpreted by the device.

**Figure C-1.10 Symbolic Segment Encoding**

Character Symbols can contain a maximum of 31 characters.

If the symbol is ASCII, Double-byte or Triple-byte, when comparing segments, the symbol portion shall be handled in a case insensitive manner. For instance “LS101” and “ls101” shall be considered equal.

Numeric Symbols can be 8, 16, or 32 bits.

When a Symbolic Segment is used in a Padded EPATH, if the length of the segment is odd a pad byte of 0 shall be added at the end of the segment. The size shall not include the pad byte.

For example (values are hexadecimal)**:**

**Table C-1.7 Symbolic Segment Examples**

| Symbolic Segment | Notes |
|---|---|
| [65][LS101] | ASCII Symbol. This could refer to a bit in a data structure |
| [67][Line_23] | ASCII Symbol. This could refer to a controller in a slot |
| [68][Wire_off] | ASCII Symbol. This could refer to a bit in a diagnostic structure |
| [60][22][1234][2345] | Japanese symbol |
| [60][C7][1234] | 16 bit Numeric Symbol |
| [60][C8][12345678] | 32 bit Numeric Symbol |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### C-1.4.5 Data Segment

The data segment provides a mechanism for delivering data to an application. This may occur during connection establishment, or at any other time as defined by the application.

**Figure C-1.11 Data Segment Encoding**

#### C-1.4.5.1 Simple Data Segment

The Simple data segment contains data values such as parameters for the target application. The size of the data segment is specified in number of 16 bit words and depends on the application. This byte value immediately follows the segment type. The data values follow the length byte. The data segment can be any number of 16 bit words up to 255. An encoding can contain more than one data segment if required.

The following table provides examples of data segments:

**Table C-1.8 Data Segment Encoding**

| Encoding | Notes |
|---|---|
| [80][07] [0100] [0200] [0300] [0400] [0500] [0600] [0700] | Seven words of data in a data segment |
| [21] [0500] [24] [09] [80][04] [0100] [0200] [0300] [0400] | Logical segments for Class 5, Instance 9, then four words of data in a data segment |

The data can be configuration information for the object, additional parameters necessary for the object, or any other set of object specific information.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### C-1.4.5.2 ANSI Extended Symbol Segment

The segment type of ANSI extended symbol segment shall be 0x91. The byte following the segment type (second byte) shall represent the number of characters (8-bit) in the symbol (symbol size). The variable length symbol shall follow the symbol size and shall end with a pad byte of zero if the size is of odd length. The symbol size shall not count the pad byte if it is included.

The character portion of the ANSI extended symbol segment shall be handled in a case insensitive manner. For instance "start1" and "Start1" shall be considered equal.

The following table provides examples of extended symbol segments:

**Table C-1.9 ANSI Extended Symbol Segment Examples**

| Encoding | Notes |
|---|---|
| [91][06] [73] [74] [61] [72] [74] [31] | Six bytes of data in the symbol 'start1' |
| [91] [07] [73] [74] [61][72] [74] [65] [72] [00] | Seven bytes of data in the symbol 'starter' (plus a null pad) |

## C-1.5 Segment Definition Hierarchy

In general, the definition of any rules related to the use of segments is defined within the Object Class and/or Device Profile. When symbolic and/or logical segments are used to construct an application path the Backus-Naur Form definition is:

*application_path* ::= CHOICE {*symbolic_application_path, class_application_path, assembly_class_application_path*}

*symbolic_application_path* ::= *symbolic_segment* [Connection_Point] [*member_specification*] [*bit_specification*]

*symbolic_segment* ::= CHOICE {Symbolic_Segment, ANSI_Extended_Symbol_Segment}

*assembly_class_application_path ::= 20 04* [*assembly_attribute_specification*]

*class_application_path* ::= Class_ID[1] [*item_specification*]

*item_specification* ::= CHOICE {*attribute_specification, connection_point_specification*}

*assembly_attribute_specification* ::= CHOICE {Instance ID, Connection Point} [[Attribute ID] [*member_specification*] [*bit_specification*]]

*attribute_specification* ::= Instance ID [[Attribute ID] [*member_specification*] [*bit_specification*]]

*connection_point_specification* ::= [Instance ID] Connection Point [*member_specification*] [*bit_specification*]

*member_specification* ::= CHOICE {Member_ID, *extended_member_specification*}

*extended_member_specification* ::= SEQUENCE of {Array_Index, *indirect_array_specification*, Structure_Member_Number, Structure_Member_Handle}

*indirect_array_specification* ::= Indirect_Array_Index *application_path*

*bit_specification* ::= CHOICE {Bit_Index, *indirect_index_bit_specification*}

*indirect_bit_specification* ::= Indirect_Bit_Index *application_path*

[1] The Class ID cannot specify the Assembly Object (20 04).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

NOTE: A *Symbolic_application_path* must resolve, at run time, to a *Class_application_path*.

NOTE: An Indirect_Array_Index and Indirect_Bit_Index *application_path* must evaluate to a positive integer.

The depth within the application path need only proceed to the degree required by its application.

The following are examples of valid application paths.

- Class ID,
- Class ID, Instance ID
- Class ID, Instance ID, Attribute ID
- Class ID, Instance ID, Attribute ID, Member ID
- Class ID, Connection Point
- Class ID, Connection Point, Member ID
- Class ID, Instance ID, Connection Point[2]
- Class ID, Instance ID, Connection Point, Member ID [2]
- Symbolic ID
- Symbolic ID, Member ID
- Symbolic ID, Connection Point [3]
- Symbolic ID, Connection Point, Member ID [3]
- Symbolic ID, Array Index, Structure Member Number, Array Index, Bit Index
- Symbolic ID, Indirect_Array_Index, [Class_ID, Instance_ID, Attribute_ID], Structure_Member_Number, Array_Index, Bit_Index

The following example demonstrates encoding for two path attributes: the **produced_connection_path** and **consumed_connection_path** attributes of the Connection Object.

[2] Instance ID and Connection Point for Class ID 4 (Assembly Object) shall not be used together, since for the Assembly object they are defined to be the same.

[3] If the Symbolic ID resolves to a Class ID of 4 (Assembly Object) and Instance ID then Symbolic ID and Connection Point shall not be used together, since for the Assembly object Instance and Connection Point are defined to be the same.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Figure C-1.12 Example Encoding for Connection Object Paths**

## C-1.6 Encoded Path Compression Rules

When multiple encoded paths are concatenated the delineation between paths is where a segment at a higher level in the hierarchy is encountered. Multiple encoded paths may be compacted when each path shares the same values at the higher levels in the hierarchy. Extended Logical Segments shall not be used in compressed paths. When a segment is encountered which is at the same or higher level but not at the top level in the hierarchy, the preceding higher levels are used for that next encoded path. The examples below show multiple encoded paths in the full and compacted representations.

Full: Class A, Instance A, Attribute A, Class A, Instance A, Attribute B
Compact: Class A, Instance A, Attribute A, Attribute B

Full: Class A, Instance A, Attribute A, Class A, Instance B, Attribute A
Compact: Class A, Instance A, Attribute A, Instance B, Attribute A


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# C-2 Data Type Specification

This section describes the data type specification syntaxes, data type value ranges, and operations that can be performed on the defined data types. The specification of a data type comprises:

- the range of values that variables of the type may take on, and
- the operations performed on these variables

This CIP standard specifies *elementary* and *derived* data types corresponding to the notation of IEC 1131-3. In addition, since function blocks as defined in IEC 1131-3 have both an associated data structure and a set of defined standard operations, these elements are also specified as data types in this CIP standard.

## C-2.1 Data Type Values

Data is made up of ***elementary*** (primitive) data types. These elementary data types are used to construct ***derived*** (constructed) data types.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### C-2.1.1 Elementary Data Types

The elementary data types and the values (range) of variables of each type are given in Table C-2.1, Elementary Data Types and its associated notes. This table specifies the values of certain parameters which are identified as “implementation–dependent” in Table 10.1 of subclause 2.3.1 in IEC 1131-3. The implementation–dependent parameters defined in this specification can be found in Table C-3.4, Values of Implementation–dependent Parameters.

**Table C-2.1 Elementary Data Types**

| Keyword | Description | Range | |
|---|---|---|---|
| | | Minimum | Maximum |
| BOOL | Boolean | NOTE 1 | |
| SINT | Short Integer | -128 | 127 |
| INT | Integer | -32768 | 32767 |
| DINT | Double Integer | $-2^{31}$ | $2^{31}-1$ |
| LINT | Long Integer | $-2^{63}$ | $2^{63}-1$ |
| USINT | Unsigned Short Integer | 0 | 255 |
| UINT | Unsigned Integer | 0 | 65535 |
| UDINT | Unsigned Double Integer | 0 | $2^{32}-1$ |
| ULINT | Unsigned Long Integer | 0 | $2^{64}-1$ |
| REAL | Floating point | NOTE 2 | |
| LREAL | Long float | NOTE 3 | |
| ITIME | Duration (short) | NOTE 12 | |
| TIME | Duration | NOTE 4 | |
| FTIME | Duration (high resolution) | NOTE 5,6 | |
| LTIME | Duration (long) | NOTE 6,7 | |
| DATE | Date only | NOTE 8 | |
| TIME_OF_DAY or TOD | Time of day | NOTE 9 | |
| DATE_AND_TIME or DT | Date and time of day | NOTE 10 | |
| STRING | character string (1 byte per character) | | |
| STRING2 | character string (2 bytes per character) | NOTE 6 | |
| STRINGN | character string (N-bytes per character) | NOTE 6 | |
| SHORT_STRING | character string (1 byte per character, 1 byte length indicator) | NOTE 6 | |
| STRINGI | International character string | NOTE 6 | |
| BYTE | bit string - 8 bits | NOTE 11 | |
| WORD | bit string - 16-bits | NOTE 11 | |
| DWORD | bit string - 32-bits | NOTE 11 | |
| LWORD | bit string - 64-bits | NOTE 11 | |
| EPATH | CIP path segments | NOTE 13 | |
| ENGUNIT | Engineering Units | NOTE 14 | |

1 The possible values of variables of type BOOL 0 and 1, corresponding to the keywords FALSE and TRUE, respectively.

2 The range of values for variables of type REAL are defined in IEEE 754 for the basic single floating-point format.

3 The range of values for variables of type LREAL are defined in IEEE 754 for the basic double floating-point format.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Keyword | Description | Range | |
|---|---|---|---|
| | | Minimum | Maximum |
| 4 | The range of values for variables of type TIME is the same as for variables of type DINT, representing the time duration in milliseconds, i.e., a range of T#-24d20h31m23.648s to T#24d20h31m23.647s. | | |
| 5 | The range of values for variables of type FTIME is the same as for variables of type DINT, representing the time duration in microseconds, i.e., a range of T#-35m47.483648s to T#35m47.483647s. | | |
| 6 | This is a CIP standard extension to IEC 1131-3. | | |
| 7 | The range of values for variables of type LTIME is the same as for variables of type LINT, representing the time duration in microseconds, i.e., a range of T#-106751991d4h0m54.775808s to T#106751991d4h0m54.775807s. | | |
| 8 | The range of values for variables of type DATE is from D#1972-01-01, the start of the Coordinated Universal Time (UTC) era, to D#2151-06-06 (a total range of 65536 days). | | |
| 9 | The range of values for variables of type TIME_OF_DAY is from TOD#00:00:00.000 to TOD#23:59:59.999 to a resolution of 1 millisecond. | | |
| 10 | The range of values for variables of type DATE_AND_TIME is from DT#1972-01-01-00:00:00.000 to DT#2151-06-06-23:59:59.999. | | |
| 11 | Values of bit string data types is in the range $2\#b_{N-1}b_{N-2}...b_2b_1b_0$, where N is the number of bits in the bit string, $b_{N-1}$ is the "most significant bit", and $b_0$ is the "least significant bit". The value of the j-th bit $b_j$ is represented as 0 or 1, corresponding to the Boolean value FALSE or TRUE, respectively. | | |
| 12 | The range of values for variables of type ITIME is the same as for variables of type INT, representing the time duration in milliseconds, i.e., a range of T#-32s768ms to T#32s767ms. | | |
| 13 | For complete information on the EPATH data type, refer to Appendix C: Abstract Syntax Encoding for Segment Types. | | |
| 14 | The range of values for variables of type ENGUNIT is the same as for variables UINT, representing the values enumerated in Appendix D of the CIP specification. | | |

### C-2.1.2 Character String Data Types

The declaration of a variable of type STRING, STRING2, or STRINGN is equivalent to declaring a structured data type for the variable which allocates a UINT variable containing the current size of the string in characters and an array of declared character size elements. STRING declares 8-bit octet elements. STRING2 declares 16-bit octet elements. STRINGN declares N*8-bit octets. STRINGN includes a UINT variable which declares the character size. Character strings of data type STRING shall be based on ISO-8859-1. Character strings of data type STRING2 and STRINGN shall be based on ISO 10646.

The declaration of a variable of type SHORT_STRING is equivalent to declaring a structured data type for the variable which allocates a USINT variable containing the current size of the string in characters and an array of 8-bit octet elements. Character strings of this data type shall be based on ISO-8859-1.

The declaration of a variable of type STRINGI allows for multiple language representation for a single string.

The structure of a STRINGI is defined in the Table C-2.2.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table C-2.2 STRINGI Definition**

| Name | Data Type | Meaning |
|---|---|---|
| Number | USINT | The number of internationalized character strings |
| Strings | Array of :<br>Struct of : | Array of individual internationalized character strings |
| LanguageChar1 | USINT | The first ASCII character of the ISO 639-2/T language [1] |
| LanguageChar2 | USINT | The second ASCII character of the ISO 639-2/T language [1] |
| LanguageChar3 | USINT | The third ASCII character of the ISO 639-2/T language [1] |
| CharStringStruct | USINT | The structure of the character string, limited to the Elementary Data Type values 0xD0 (STRING), 0xD5 (STRING2), 0xD9 (STRINGN) and 0xDA (SHORT_STRING) |
| CharSet | UINT | The character set which the character string is based on which comes from IANA MIB Printer Codes (RFC 1759). The character set values are limited to those values that are provided in Table C-2.4 |
| InternationalString | Data type defined in CharStringStruct | An array of 8-bit octet elements which is the actual international character string |

1 The three LanguageChars for the language come from ISO 639-2/T (Alpha-3 Terminology Code). Some common language codes are shown inTable C-2.3.

SHORT_STRING, STRING2, STRINGN and STRINGI are extensions to IEC 1131-3.

**Table C-2.3 Common Language Codes from ISO 639-2/T**

| Language | First Character (language1) | Second Character (language2) | Third Character (language3) |
|---|---|---|---|
| English | e | n | g |
| French | f | r | a |
| Spanish | s | p | a |
| Italian | i | t | a |
| German | d | e | u |
| Japanese | j | p | n |
| Portuguese | p | o | r |
| Chinese | z | h | o |
| Russian | r | u | s |

**Table C-2.4 Valid IANA MIB Printer Codes for Character Set Selection**

| Character Set | Value |
|---|---|
| ISO-8859-1:1987 | 4 |
| ISO-8859-2:1987 | 5 |
| ISO-8859-3:1988 | 6 |
| ISO-8859-4:1988 | 7 |
| ISO-8859-5:1988 | 8 |
| ISO-8859-6:1987 | 9 |
| ISO-8859-7:1987 | 10 |
| ISO-8859-8:1989 | 11 |
| ISO-8859-9:1989 | 12 |
| ISO-10646-UCS-2 | 1000 |
| ISO-10646-UCS-4 | 1001 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### C-2.1.3 Structured Bit String Types

Structured bit string types are a CIP extension of the derived data type mechanisms specified in subclause 2.2.3 of IEC 1131-3. These data types are based on the IEC standard bit string types (ANY_BIT = BYTE, WORD, DWORD, or LWORD).

### C-2.1.4 Derived Data Types

The derived data types specified by CIP are:

- directly derived
- enumerated
- subrange
- structured
- array

These data types are defined in subclauses 1.3 and 2.3.3 of IEC 1131-3. The means of specifying these data types and their default initial values is defined in subclauses 2.3.3.1 and 2.3.3.2 of IEC 1131-3. The usage of variables of these data types is defined in subclause 2.3.3.3 of IEC 1131-3.

### C-2.1.5 Defined Constructed Data Types

Defined Constructed Data Types are data types whose structure is predefined and usage is common throughout the CIP Networks Library.

**Table C-2.5 Defined Constructed Data Types**

| Keyword | Description |
|---|---|
| SIGNED_ODOMETER | STRUCT of $x10^{n}$, $x10^{n+3}$, $x10^{n+6}$, $x10^{n+9}$, $x10^{n+12}$<br>Signed large magnitude counter. See section C-2.1.5.1 |
| ODOMETER | STRUCT of $x10^{n}$, $x10^{n+3}$, $x10^{n+6}$, $x10^{n+9}$, $x10^{n+12}$<br>Unsigned large magnitude counter. See section C-2.1.5.2 |
| ETH_MAC_ADDR | Ethernet MAC Address. See section C-2.1.5.3 |

#### C-2.1.5.1 SIGNED_ODOMETER Defined Constructed Data Type

The SIGNED_ODOMETER data type signifies a signed large magnitude counter capable of representing a range of values from -999,999,999,999,999 to 999,999,999,999,999.

**Table C-2.6 SIGNED_ODOMETER Defined Constructed Data Type**

| Name | Data Type Structure | Description of Data Type Element | Semantics of Value |
|---|---|---|---|
| SIGNED_ODOMETER | STRUCT of: | | |
| | INT | $x10^{n}$ | ±Unit[1] x $10^{n}$ |
| | INT | $x10^{n+3}$ | ±Unit[1] x $10^{n+3}$ |
| | INT | $x10^{n+6}$ | ±Unit[1] x $10^{n+6}$ |
| | INT | $x10^{n+9}$ | ±Unit[1] x $10^{n+9}$ |
| | INT | $x10^{n+12}$ | ±Unit[1] x $10^{n+12}$ |

Table Footnotes

1. Unit can be any engineering unit defined in Appendix D for example, Energy represented in kWh.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Each signed integer in the SIGNED_ODOMETER data type structure shall contain values from -999 to 999. To represent a positive SIGNED_ODOMETER value, all members of the structure shall be positive or zero. To represent a negative SIGNED_ODOMETER value, all members of the structure shall be negative or zero. When all members of the structure are zero, the SIGNED_ODOMETER value is neither positive nor negative.

The units of the SIGNED_ODOMETER structure can be any engineering unit and the value of n is used to determine the location of a decimal point. The value of n shall be defined where the usage of the SIGNED_ODOMETER type is specified (e.g. in the semantics of an attribute). The valid range of n shall be a SINT between 0 and -15.

For example in the table below if the SIGNED_ODOMETER represented Kilowatt hours and n = -3.

**Table C-2.7 SIGNED_ODOMETER Kilowatt hours units and n = -3**

| $x10^{n+12}$ | $x10^{n+9}$ | $x10^{n+6}$ | $x10^{n+3}$ | $x10^{n}$ |
|---|---|---|---|---|
| Terawatt-hours ($kWh \times 10^{9}$) | Gigawatt-hours ($kWh \times 10^{6}$) | Megawatt-hours ($kWh \times 10^{3}$) | Kilowatt-hours (kWh) | Watt-hours ($kWh \times 10^{-3}$) |

Consider the following example using kWh as the units and n = -3. The positive value of 123,456,789.012 kWh would be represented as follows:

**Table C-2.8 SIGNED_ODOMETER Kilowatt hours units, n = -3, positive value example**

| $x10^{n+12}$ | $x10^{n+9}$ | $x10^{n+6}$ | $x10^{n+3}$ | $x10^{n}$ |
|---|---|---|---|---|
| 0 | 123 | 456 | 789 | 12 |

Consider the following example using kWh as the units and n=-3. The negative value of -789,456,123.045 kWh would be represented as follows:

**Table C-2.9 SIGNED_ODOMETER Kilowatt hours units, n = -3, negative value example**

| $x10^{n+12}$ | $x10^{n+9}$ | $x10^{n+6}$ | $x10^{n+3}$ | $x10^{n}$ |
|---|---|---|---|---|
| 0 | -789 | -456 | -123 | -45 |

For a positive SIGNED_ODOMETER value, when a structure member at 999 is incremented by 1, the member value shall go to 0 and member+1 value increments by 1 (e.g. 1,999 goes to 2,000). If no member+1 value exists, the overflow is simply discarded. Hence, if a SIGNED_ODOMETER at 999,999,999,999,999 is incremented by 1, the SIGNED_ODOMETER value shall go to 0. For a non-zero positive SIGNED_ODOMETER value, if a structure member at 0 is decremented by 1, the member value shall go to 999 and member+1 value is decremented by 1 (e.g. 1,000,000 goes to 999,999).

Note that decrementing a SIGNED_ODOMETER that is at a value of zero is a special case (e.g. 0 goes to -6 if decremented by 6).

For a SIGNED_ODOMETER with a negative value, when a member value at -999 is decremented, the member value shall go to 0 and member+1 value decrements by 1 (e.g. -123,456,999 goes to -123,457,000). If no member+1 value exists, the underflow is simply discarded. Hence, if a SIGNED_ODOMETER at -999,999,999,999,999 is decremented by 1, it shall go to 0. For a negative SIGNED_ODOMETER value, if a structure member at 0 is incremented by 1, the member value shall go to -999 and the member+1 value increments by 1 (e.g. –111,222,000 goes to -111,221,999).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The following examples show mathematical and comparison operations and results for the SIGNED_ODOMETER data type.

999,999,999,999,992 + 27 = 19

-999,999,999,999,988 – 42 = -30

144,988.012 – 200,000.000 = -55,011.988

-180.017 + 300.0 = 119.983

999,999,999,999,997 > -999,999,999,999,999

-627,147 < 123,456

In a device, updating a SIGNED_ODOMETER value will most likely happen based on sampling some data at some periodic rate, and as a consequence (due to long sampling times or fast data rates), it may change by more than 1 or -1 at an update. Regardless of the method used in a device to implement a SIGNED_ODOMETER (e.g. a large integer converted to the SIGNED_ODOMETER data type, native SIGNED_ODOMETER support, etc.) updating a SIGNED_ODOMETER can be a time-consuming activity. In addition, external devices will need to access SIGNED_ODOMETER values, either by explicit or implicit messaging. To preserve the complete value of the SIGNED_ODOMETER, access to the SIGNED_ODOMETER data structure shall be atomic, i.e. access to individual members is not allowed. Implementations may want to consider using temporary SIGNED_ODOMETER values for time-consuming operations followed by mutually-exclusive memory copy operations to minimize mutual exclusion lockout times.

### C-2.1.5.2 ODOMETER Defined Constructed Data Type

The ODOMETER data type signifies an unsigned large magnitude counter capable of representing a range of values from 0 to 999,999,999,999,999.

**Table C-2.10 ODOMETER Defined Constructed Data Type**

| Name | Data Type Structure | Description of Data Type Element | Semantics of Value |
|---|---|---|---|
| ODOMETER | STRUCT of: | | |
| | UINT | $x10^{n}$ | ±Unit[1] x $10^{n}$ |
| | UINT | $x10^{n+3}$ | ±Unit[1] x $10^{n+3}$ |
| | UINT | $x10^{n+6}$ | ±Unit[1] x $10^{n+6}$ |
| | UINT | $x10^{n+9}$ | ±Unit[1] x $10^{n+9}$ |
| | UINT | $x10^{n+12}$ | ±Unit[1] x $10^{n+12}$ |

Table Footnotes

1. Unit can be any engineering unit defined in Appendix D for example, Energy represented in kWh.

Each unsigned integer in the ODOMETER data type structure shall contain values from 0 to 999. To represent an ODOMETER value, all members of the structure shall be positive or zero.

The units of the ODOMETER structure can be any engineering unit and the value of n is used to determine the location of a decimal point. The value of n shall be defined where the usage of the ODOMETER type is specified (e.g. in the semantics of an attribute). The valid range of n shall be a SINT between 0 and -15.

For example in the table below if the ODOMETER represented meters m and n = -9.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table C-2.11 ODOMETER meters units and n = -9**

| $x10^{n+12}$ | $x10^{n+9}$ | $x10^{n+6}$ | $x10^{n+3}$ | $x10^{n}$ |
|---|---|---|---|---|
| 1000 cM x 1012 | 1000 cM x 109 | 1000 cM x 106 | 1000 cM x 103 | 1000 cM x 100 |

Using the above units, a value of 123.456789012 m would be represented as:

**Table C-2.12 ODOMETER meters units and n = -9 example**

| $x10^{n+12}$ | $x10^{n+9}$ | $x10^{n+6}$ | $x10^{n+3}$ | $x10^{n}$ |
|---|---|---|---|---|
| 0 | 123 | 456 | 789 | 12 |

For an ODOMETER value, when a structure member at 999 is incremented by 1, the member value shall go to 0 and member+1 value increments by 1 (e.g. 1,999 goes to 2,000). If no member+1 value exists, the overflow is simply discarded. Hence, if an ODOMETER at 999,999,999,999,999 is incremented by 1, the ODOMETER value shall go to 0.For an ODOMETER value, if a structure member at 0 is decremented by 1, the member value shall go to 999 and member+1 value is decremented by 1 (e.g. 1,000,000 goes to 999,999). If no member+1 value exists, the underflow is simply discarded. Hence, if an ODOMETER at 0 is decremented by 1, the ODOMETER value shall go to 999,999,999,999,999.

The following examples show mathematical operations and results for the ODOMETER data type.

999,999,999,999,992 + 27 = 19

144,988.012 – 200,000.000 = 999,999,944,988.012

In a device, updating an ODOMETER value will most likely happen based on sampling some data at some periodic rate, and as a consequence (due to long sampling times or fast data rates), it may change by more than 1 or -1 at an update. Regardless of the method used in a device to implement an ODOMETER (e.g. a large integer converted to the ODOMETER data type, native ODOMETER support, etc.) updating an ODOMETER can be a time-consuming activity. In addition, external devices will need to access ODOMETER values, either by explicit or implicit messaging. To preserve the complete value of the ODOMETER, access to the ODOMETER data structure shall be atomic, i.e. access to individual members is not allowed. Implementations may want to consider using temporary ODOMETER values for time-consuming operations followed by mutually-exclusive memory copy operations to minimize mutual exclusion lockout times.

### C-2.1.5.3 ETH_MAC_ADDR Defined Constructed Data Type

**Table C-2.13 ETH_MAC_ADDR Defined Constructed Data Type**

| Name | Data Type Structure | Description of Data Type Element | Semantics of Value |
|---|---|---|---|
| ETH_MAC_ADDR | Array of 6 octets | Ethernet MAC Layer Address | |

The Ethernet MAC Address is an array of 6 octets. The recommended display format is “XX-XX-XX-XX-XX-XX”, starting with the first octet. The Ethernet MAC Address is assigned by the manufacturer and shall be unique per IEEE 802.3 requirements. The general requirement is that the value of this Type shall be Ethernet MAC addresses used in Ethernet frames.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# C-3 IEC 1131-3 Compliance

Subclause 1.5.1 of IEC 1131-3 defines the requirements, which are met by programmable controller systems claiming compliance to the language standard. This provides the data type-related information to be included in the documentation of CIP functional units, which support the data types defined in this CIP standard. Subsets or extensions of this documentation are provided as appropriate to the specific compliant functional unit.

This section provides information with respect to only the data types and associated operations defined in this CIP standard. For full compliance with IEC 1131-3, documentation of the additional language elements supported by the functional unit must be provided as prescribed in subclause 1.5.1 of IEC 1131-3.

Included in this section are the following:

- Compliance Statement
- Implementation-Dependent Parameters
- Error Conditions
- Language Extensions
- Formal Syntax

## C-3.1 Compliance Statement

This system complies with the requirements of IEC 1131-3 for the following language features:

**Table C-3.1 Common Elements**

| Table/ Feature | Feature description |
|---|---|
| 10/1 | BOOL data type |
| 10/2 | SINT data type |
| 10/3 | INT data type |
| 10/4 | DINT data type |
| 10/5 | LINT data type |
| 10/6 | USINT data type |
| 10/7 | UINT data type |
| 10/8 | UDINT data type |
| 10/9 | ULINT data type |
| 10/10 | REAL data type |
| 10/11 | LREAL data type |
| 10/12 | TIME data type |
| 10/13 | DATE data type |
| 10/14 | TIME_OF_DAY or TOD data type |
| 10/15 | DATE_AND_TIME or DT data type |
| 10/16 | STRING data type |
| 10/17 | BYTE data type |
| 10/18 | WORD data type |
| 10/19 | DWORD data type |
| 10/20 | LWORD data type |
| 12/1 | Directly derived data types |
| 12/2 | Enumerated data types |
| 12/3 | Subrange data types |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Table/ Feature | Feature description |
|---|---|
| 12/4 | Array data types |
| 12/5 | Structured data types |
| 13 | Standard default initial values |
| Subclause 2.5.1.3 | User-declared functions (no table entry) |
| 22/1 | Type conversions (see Table C-3.3) |
| 22/2 | TRUNC function |
| 22/3 | BCD_TO_** functions |
| 22/4 | *_TO_BCD functions |
| 23/1-11 | Standard functions of one numeric variable: ABS, SQRT, LN, LOG, EXP, SIN, COS, TAN, ASIN, ACOS, ATAN |
| 24/12n-18n | Standard named arithmetic functions: ADD, MUL, SUB, DIV, MOD, EXPT, MOVE |
| 24/12s-15s, 17s, 18s | Standard symbolic arithmetic functions:<br>+, *,–, /, **, := |
| 25/1-4 | Standard bit string functions: SHL, SHR, ROR, ROL |
| 26/5s-8s | Standard named bitwise Boolean functions:<br>AND, OR, XOR, NOT |
| 26/5s-7s | Standard symbolic bitwise Boolean functions:<br>&, >=1, =2k+1 |
| 27/1-4 | Standard selection functions: SEL, MAX, MIN, LIMIT, MUX |
| 28/5n-10n | Standard named comparison functions: GT, GE, EQ, LE, LT, NE |
| 28/5s-10s | Standard symbolic comparison functions: >, >=, =, <=, <, <> |
| 29/1-9 | Standard character string functions: LEN, LEFT, RIGHT, MID, CONCAT, INSERT, DELETE, REPLACE, FIND |
| 30/1-14 | Standard functions of time data types: ADD, SUB, MUL, DIV, CONCAT, DATE_AND_TIME_TO_TIME_OF_DAY, TIME_OF_DAY_TO_DATE_AND_TIME<br>NOTE– Table 30 of IEC 1131-3 limits the data types to which these operations apply. |
| 31/1-4 | Standard functions of enumerated data types: SEL, MUX, EQ, NE |
| 32 | Standard access mechanisms to function block I/O parameters |
| 33/8a, 8b, 9a, 9b | User-defined function blocks per subclause 2.5.2.2, with graphical or textual rising or falling edge input options |
| 34/1-3 | Standard bistable function blocks: SR, RS, SEMA |
| 35/1,2 | Standard edge detection function blocks: R_EDGE, F_EDGE |
| 36/1-3 | Standard counter function blocks: CTU, CTD, CTUD |
| 37/1,2a, 3a, 4 | Standard timer function blocks: TP, TON, TOF, RTC |
| 55/1-17 | Standard operators: (), function evaluation, **,–, NOT, *, /, MOD, +,–, <, >, <=, >=, =, <>, &, AND, XOR, OR |

**Table C-3.2 ST Language Elements**

| Table/ Feature | Feature description |
|---|---|
| 55/1-17 | Standard operators: (), function evaluation, **,–, NOT, *, /, MOD, +,–, <, >, <=, >=, =, <>, &, AND, XOR, OR |
| 56/2 | Function block invocation and output usage |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Table C-3.3 Type Conversion Operations**

Refer to the notes following this table.

| Operation | Result | Error conditions |
|---|---|---|
| ANY_BIT_TO_ANY_BIT | (NOTE 4) | None |
| ANY_BIT_TO_ANY_INT | $OUT_{min} + Sb_k2^k$ (NOTE 5) | Result > OUTmax |
| ANY_BIT_TO_BOOL | FALSE if IN = 0<br>TRUE otherwise | None |
| ANY_BIT_TO_STRING | (NOTE 6) | None |
| ANY_DATE_TO_STRING | (NOTE 7) | None |
| ANY_INT_TO_BOOL | FALSE if IN = 0<br>TRUE otherwise | None |
| ANY_NUM_TO_ANY_INT | IN (NOTE 8) | (IN > $OUT_{max}$) or<br>(IN < $OUT_{min}$) |
| ANY_NUM_TO_ANY_REAL | IN | (NOTE 9) |
| ANY_NUM_TO_DATE | (NOTE 10) | |
| ANY_NUM_TO_STRING | (NOTE 11) | |
| ANY_NUM_TO_TIME | (NOTE 12) | |
| ANY_NUM_TO_TOD | (NOTE 13) | |
| ANY_REAL_TO_BOOL | FALSE if IN = 0.0<br>TRUE otherwise | None |
| BOOL_TO_ANY_BIT<br>BOOL_TO_ANY_INT | 0 if IN = FALSE<br>1 if IN = TRUE | None |
| BOOL_TO_ANY_REAL | 0.0 if IN = FALSE<br>1.0 if IN = TRUE | None |
| BOOL_TO_STRING | ’FALSE’ if IN = FALSE<br>’TRUE’ if IN = TRUE | None |
| DATE_TO_ANY_NUM | (NOTE 14) | Result > $OUT_{max}$ |
| STRING_TO_ANY | Converted data | (NOTE 15) |
| TIME_TO_ANY_NUM | (NOTE 16) | Result > $OUT_{max}$ |
| TOD_TO_ANY_NUM | (NOTE 17) | Result > $OUT_{max}$ |

| | |
|---|---|
| 1 | Use of the generic data types ANY_NUM, ANY_REAL, ANY_INT, and ANY_BIT defined in subclause 2.3.2 of IEC 1131-3 is intended to imply a family of conversions. For instance, the conversion BOOL_TO_ANY_REAL is intended to imply BOOL_TO_REAL and BOOL_TO_LREAL. |
| 2 | IN refers to the value of the input variable of the type conversion function. |
| 3 | $OUT_{min}$ and $OUT_{max}$ refer to the minimum and maximum values of the output data type of the conversion function, as defined in Table C-2.1. |
| 4 | In conversions of bit string types, if the number of bits in the input variable IN is less than the number of the bits in the output variable OUT, the bits of the input is copied to the corresponding least significant bits of the result and the remainder of the result is zero-filled. If the number of bits of IN is greater than the number of bits of OUT, the least significant bits of IN is copied to the corresponding bits of the result. For instance,<br>BYTE_TO_WORD(16#FF) = 16#00FF<br>and<br>WORD_TO_BYTE(16#0FF0) = 16#F0 |
| 5 | Bit numbering in this expression is as specified in Note 11 of Table C-2.1. |
| 6 | The result of conversion of a bit string variable to type STRING consists of a string containing the base 16 literal representation of the variable value, as defined in subclause 2.2.1 of IEC 1131-3, in characters taken from the ISO 646 character set. |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| | Operation | Result | Error conditions |
|---|---|---|---|
| 7 | The result of conversion of a date and/or time of day variable to type STRING consists of a string containing the literal representation of the variable value, as defined in subclause 2.2.1 of IEC 1131-3, in characters taken from the ISO 646 character set. | | |
| 8 | Conversion of REAL and LREAL to integer types is accomplished by rounding as specified in subclause 5.4 of IEEE 754. | | |
| 9 | Rounding errors may occur if the number of significant bits in the input variable is larger than the number of significant bits in the output floating-point representation. Also, range errors of the type noted for ANY_NUM_TO_ANY_INT may occur in LREAL_TO_REAL. | | |
| 10 | Conversion of a variable of a numeric type to DATE has the same result as conversion of the variable to UINT, with the result being interpreted as the number of days since 1972-01-01. | | |
| 11 | Conversion of a variable of a numeric type to type STRING consists of a string containing the literal representation of the variable value, as defined in subclause 2.2.1 of IEC 1131-3, in characters taken from the ISO 646 character set. | | |
| 12 | Conversion of a variable of a numeric type to TIME has the same result as conversion of the variable to DINT, with the result being interpreted as a duration in milliseconds. | | |
| 13 | Conversion of a variable of a numeric type to TOD has the same result as conversion of the variable to UDINT, with the result being interpreted as a time since midnight in milliseconds. | | |
| 14 | Conversion of a variable of type DATE to a numerical type is the same as the conversion of a variable of type UINT to the corresponding numerical type, with the result being the numerical equivalent of the days since 1972-01-01. | | |
| 15 | It is an error if the STRING data to be converted is not in the format for external representation of the output data type as specified in subclause 2.2 of IEC 1131-3, or if the result of the conversion is outside the range $\{OUT_{min}..OUT_{max}\}$. | | |
| 16 | Conversion of a variable of type TIME to a numerical type is the same as the conversion of a variable of type DINT to the corresponding numerical type, with the result being the numerical equivalent of the corresponding time interval expressed in milliseconds. | | |
| 17 | Conversion of a variable of type TOD (TIME_OF_DAY) to a numerical type is the same as the conversion of a variable of type UDINT to the corresponding numerical type, with the result being the numerical equivalent of the time since midnight expressed in milliseconds. | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## C-3.2 Implementation-Dependent Parameters

Table C-3.4 lists the values of implementation-dependent parameters from Table D.1 of IEC 1131-3 that are defined in this CIP standard. Values of other implementation-dependent parameters are defined in other CIP standards or in the specifications of individual functional units as appropriate.

**Table C-3.4 Values of Implementation–dependent Parameters**

| Clause of IEC 1131-3 | Parameter | Value |
|---|---|---|
| 2.2.3.1 | Range of values of duration | Same as LINT in microseconds |
| 2.3.1 | Range of values for variables of type TIME | Same as DINT in milliseconds |
| | Precision of representation of seconds in types TIME_OF_DAY and DATE_AND_TIME | 1 millisecond |
| 2.3.3.1 | Maximum number of enumerated values | 256 |
| 2.3.3.2 | Default maximum length of STRING variables | 256 |
| | Maximum allowed length of STRING variables | 65536 |
| 2.4.1.2 | Maximum number of subscripts | 8 |
| | Maximum range of subscript values | 0..255 |
| | Maximum number of levels of structures | 8 |
| 2.5.1.5 | Maximum inputs of extensible functions | 8 |
| 2.5.1.5.1 | Effects of type conversions on accuracy | As defined in Table C-3.3 |
| 2.5.1.5.2 | Accuracy of functions of one variable | As defined in IEEE 754 |
| 2.5.2.3.3 | PVmin, PVmax of counters | 0, 65535 |

## C-3.3 Language Extensions

The extensions to IEC 1131-3 defined in this CIP standard are listed in Table C-3.5. When these extensions are used in a particular CIP compliant functional unit, the subclause references in this table are replaced by references to the descriptions of the corresponding extensions in the functional unit's documentation.

**Table C-3.5 CIP standard extensions to IEC 1131–3**

| Subclause | Description |
|---|---|
| 2.1.1 | ITIME data type<br>FTIME data type<br>LTIME data type<br>STRING2 data type<br>STRINGN data type<br>SHORT_STRING data type<br>STRINGI data type<br>EPATH data type<br>ENGUNIT data type |
| 2.1.4 | Structured bit string types |
| 2.2 | Operations on STRING2 variables |
| 2.2.4.1 | Numbered bit string access |
| 2.2.4.2 | Structured bit string access |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# C-4 Abstract Syntax Specification

The lower layers of open system architectures are concerned with the transport of user data among distributed functional units. In these layers, the user data can be regarded simply as a sequence of octets. However, application layer entities may manipulate the values of quite complex data types. To achieve independence between the application layer and lower layers, data types are specified in an abstract syntax notation.

Supplementing the abstract syntax with one or more algorithms (called encoding rules) determines the values of the lower layer octets which carry the application layer values. The combination of the abstract syntax with a single set of transfer rules produces a specific transfer syntax.

Important: Users of this CIP standard should read and understand ISO 8824/8825, IEC 1131-3, and section C-4, Abstract Syntax Specification. Refer to these documents when reading and applying this standard.

The data type definitions provided in this CIP standard are written in Abstract Syntax Notation One (ASN.1), as defined in ISO 8824. These type definitions are part of the ASN.1 module "CIPDataTypes." The beginning ASN.1 statement indicating that these definitions are in this module is:

```
CIPDataTypes DEFINITIONS ::= BEGIN
```

and the closing ASN.1 statement is the keyword "END".

The abstract definitions that follow comprise the set of CIP Data Types. In addition, provision is made to extend or derive new data types based on existing defined types, and to include those in a "type dictionary."

## C-4.1 CIP Data Specification

The notation [typeId] for directly derived, enumerated, subrange and structured bit string data means that the tag is to be taken from the "type" field in the corresponding VariableDictionaryEntry.

```
CIPData           ::=   CHOICE{ElementaryData, DerivedData}
ElementaryData    ::=   CHOICE{
  BOOL,
  FixedLengthInteger,
  FixedLengthReal,
  AnyTime,
  AnyDate,
  AnyString,
  FixedLengthBitString,
  EPATH,
  ENGUNIT}
DerivedData   ::= CHOICE {
  DirectlyDerivedData,
  EnumeratedData,
  SubrangeData,
  StructuredBitStringData,
  ARRAY,
  STRUCT,
  FunctionBlockData}
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
DirectlyDerivedData     ::=   [typeId] CIPData
EnumeratedData          ::=   [typeId] USINT
SubrangeData            ::=   [typeId] FixedLengthInteger
StructuredBitStringData ::=   [typeId] FixedLengthBitString
FixedLengthInteger      ::=   CHOICE {SignedInteger,
                              UnsignedInteger}
SignedInteger       ::= CHOICE {SINT, INT, DINT, LINT}
UnsignedInteger     ::= CHOICE {USINT, UINT, UDINT, ULINT}
FixedLengthReal     ::= CHOICE {REAL, LREAL}
AnyTime    ::=    CHOICE {ITIME, TIME, FTIME, LTIME}
AnyDate    ::=    CHOICE {DATE, TIME_OF_DAY, DATE_AND_TIME}
AnyString  ::=    CHOICE {STRING, STRING2}
FixedLengthBitString    ::=   CHOICE {BYTE, WORD, DWORD, LWORD}
BOOL    ::= [PRIVATE 1]  IMPLICIT BOOLEAN
SINT    ::= [PRIVATE 2]  IMPLICIT OCTET STRING–- 1 octet
INT     ::= [PRIVATE 3]  IMPLICIT OCTET STRING–- 2 octets
DINT    ::= [PRIVATE 4]  IMPLICIT OCTET STRING–- 4 octets
LINT    ::= [PRIVATE 5]  IMPLICIT OCTET STRING–- 8 octets
USINT   ::= [PRIVATE 6]  IMPLICIT OCTET STRING–- 1 octet
UINT    ::= [PRIVATE 7]  IMPLICIT OCTET STRING–- 2 octets
UDINT   ::= [PRIVATE 8]  IMPLICIT OCTET STRING–- 4 octets
ULINT   ::= [PRIVATE 9]  IMPLICIT OCTET STRING–- 8 octets
REAL    ::= [PRIVATE 10] IMPLICIT OCTET STRING–- 4 octets
LREAL   ::= [PRIVATE 11] IMPLICIT OCTET STRING–- 8 octets
STIME   ::= [PRIVATE 12] IMPLICIT DINT
DATE    ::= [PRIVATE 13] IMPLICIT UINT
TIME_OF_DAY   ::=  [PRIVATE 14] IMPLICIT UDINT
DATE_AND_TIME ::=  [PRIVATE 15] IMPLICIT SEQUENCE  {
  time_of_day     UDINT,
  date            UINT }
STRING  ::= [PRIVATE 16] IMPLICIT SEQUENCE {
  Charcount  UINT,
  stringcontents OCTET STRING} -- one octet per character
BYTE    ::= [PRIVATE 17] IMPLICIT OCTET STRING–- 1 octet
WORD    ::= [PRIVATE 18] IMPLICIT OCTET STRING–- 2 octets
DWORD   ::= [PRIVATE 19] IMPLICIT OCTET STRING–- 4 octets
LWORD   ::= [PRIVATE 20] IMPLICIT OCTET STRING–- 8 octets
STRING2 ::= [PRIVATE 21] IMPLICIT SEQUENCE {
  charcount       UINT,
  string2contents OCTET STRING} –-  2 octets/ character
FTIME   ::= [PRIVATE 22] IMPLICIT DINT
LTIME   ::= [PRIVATE 23] IMPLICIT LINT
ITIME   ::= [PRIVATE 24] IMPLICIT INT
STRINGN ::= [PRIVATE 25] IMPLICIT SEQUENCE {
  charsize        UINT,
  charcount       UINT,
  stringNcontents OCTET STRING} –-  N octets/ character
SHORT_STRING ::=  [PRIVATE 26]  IMPLICIT SEQUENCE {
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
  charcount       USINT,
  stringcontents  OCTET STRING} -- one octet per character
TIME    ::= [PRIVATE 27] IMPLICIT DINT
EPATH   ::= [PRIVATE 28] IMPLICIT OCTET STRING  –- Appendix C
ENGUNIT ::= [PRIVATE 29] IMPLICIT UINT -- Appendix D
STRINGI ::= [PRIVATE 30] IMPLICIT SEQUENCE{
  Stringnum  USINT  (number of strings)
  array of   STRUCT
  language1  USINT  (first character from ISO 639-2/T)
  language2  USINT  (second character from ISO 639-2/T)
  language3  USINT  (third character from ISO 639-2/T)
  datatype   EPATH  limited to the values 0xD0,0xD5,0xD9,and 0xDA)
  charset    UINT   from IANA MIB Printer Codes (RFC 1759))
  stringcontents  CHOICE OF (SHORT_STRING, STRING, STRING2, or
                  STRINGN) -- based on datatype field
ARRAY   ::= SEQUENCE OF CIPData  –- All of same base type
STRUCT  ::= SEQUENCE OF CIPData –- May be different types
FunctionBlockData ::=   SET{
  inputs          [0]   IMPLICIT STRUCT OPTIONAL,
  outputs         [1]   IMPLICIT STRUCT OPTIONAL,
  controlInputs   [2]   IMPLICIT STRUCT OPTIONAL,
  controlOutputs  [3]   IMPLICIT STRUCT OPTIONAL}
```

## C-4.2 Data Type Specification/Dictionaries

The definition of a CIP object may include text that defines attributes. Attributes are assigned a *Data Type* in an object definition. The Data Type may be one of those defined in this appendix or may be an object specific extension to this appendix. The following definition provides a *Type Specification* for CIPData and provides a structure for extending or deriving new data types based on existing defined types.

```
Dictionary ::=    CHOICE {VariableDictionary, TypeDictionary}
VariableDictionary  ::= SEQUENCE OF VariableDictionaryEntry
VariableDictionaryEntry ::=   SEQUENCE{
  name     AnyString,
  id       FixedLengthInteger,
  type     TypeID,
  ranges   SEQUENCE OF Subrange,–- for arrays
  accessPrivilege   BOOL {READ_ONLY(0), READ_WRITE(1)}
TypeID  :=  OCTET STRING –- ASN.1 encoded tag value of the
                        -- DataTypeSpecification module
Subrange   ::=    SEQUENCE {
  minValue FixedLengthInteger,
  maxValue FixedLengthInteger}
TypeDictionary      ::= SEQUENCE OF TypeDictionaryEntry
TypeDictionaryEntry ::= SEQUENCE {
  name AnyString,
  type TypeID,
  spec DataTypeSpecification}

DataTypeSpecification ::= CHOICE {
  alt      [PRIVATE 0]  IMPLICIT AlternateTypeSpec,
  bool     [PRIVATE 1]  IMPLICIT NULL,    --    BOOL
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
  sint      [PRIVATE 2]  IMPLICIT NULL,    --    SINT
  int       [PRIVATE 3]  IMPLICIT NULL,    --    INT
  dint      [PRIVATE 4]  IMPLICIT NULL,    --    DINT
  lint      [PRIVATE 5]  IMPLICIT NULL,    --    LINT
  usint     [PRIVATE 6]  IMPLICIT NULL,    --    USINT
  uint      [PRIVATE 7]  IMPLICIT NULL,    --    UINT
  udint     [PRIVATE 8]  IMPLICIT NULL,    --    UDINT
  ulint     [PRIVATE 9]  IMPLICIT NULL,    --    ULINT
  real      [PRIVATE 10] IMPLICIT NULL,    --    REAL
  lreal     [PRIVATE 11] IMPLICIT NULL,    --    LREAL
  stime     [PRIVATE 12] IMPLICIT NULL,    --    STIME
  date      [PRIVATE 13] IMPLICIT NULL,    --    DATE
  tod       [PRIVATE 14] IMPLICIT NULL,    --    TIME_OF_DAY
  dat       [PRIVATE 15] IMPLICIT NULL,    --    DATE_AND_TIME
  str1      [PRIVATE 16] IMPLICIT NULL,    --    STRING
  byte      [PRIVATE 17] IMPLICIT NULL,    --    BYTE
  word      [PRIVATE 18] IMPLICIT NULL,    --    WORD
  dword     [PRIVATE 19] IMPLICIT NULL,    --    DWORD
  lword     [PRIVATE 20] IMPLICIT NULL,    --    LWORD
  str2      [PRIVATE 21] IMPLICIT NULL,    --    STRING2
  ftime     [PRIVATE 22] IMPLICIT NULL,    --    FTIME
  ltime     [PRIVATE 23] IMPLICIT NULL,    --    LTIME
  itime     [PRIVATE 24] IMPLICIT NULL,    --    ITIME
  strN      [PRIVATE 25] IMPLICIT NULL,    --    STRINGN
  shstr     [PRIVATE 26] IMPLICIT NULL,    --    SHORT_STRING
  time      [PRIVATE 27] IMPLICIT NULL,    --    TIME
  epath     [PRIVATE 28] IMPLICIT NULL,    --    EPATH
  engunit   [PRIVATE 29] IMPLICIT NULL,    --    ENGUNIT
  strI      [PRIVATE 30] IMPLICIT NULL,    --    STRINGI
  constructedData CHOICE {
       abbrvStruc [0]   IMPLICIT    AbbreviatedStrucTypeSpec,
       abbrvArr   [1]   IMPLICIT    AbbreviatedArrayTypeSpec,
       frmlStruc  [2]   IMPLICIT    FormalStrucTypeSpec,
       frmlArr    [3]   IMPLICIT    FormalArrayTypeSpec,
       expBitStr  [4]   IMPLICIT    ExpandedFixedLenBitStrTypeSpec,
       expStr1    [5]   IMPLICIT    ExpandedStringTypeSpec,
       expStr2    [6]   IMPLICIT    ExpandedString2TypeSpec,
       frm1HandleStruc [7] IMPLICIT FormalHandleStrucTypeSpec }
}
AbbreviatedStrucTypeSpec      ::=   UINT
AbbreviatedArrayTypeSpec      ::=   DataTypeSpecification
FormalStrucTypeSpec ::= SEQUENCE OF DataTypeSpecification
FormalHandleStrucTypeSpec ::= SEQUENCE OF
DataTypeHandleSpecification
DataTypeHandleSpecification ::= DataTypeSpecification MemberHandle
MemberHandle ::= USINT, UINT, UDINT
FormalArrayTypeSpec ::= SEQUENCE {
  numberOfElements IMPLICIT FixedLengthInteger, - Array Number of
Elements
  dataType   DataTypeSpecification }
ExpandedFixedLenBitStrTypeSpec      ::=   SEQUENCE {
  bitStrType DataTypeSpecification -- BYTE, WORD, DWORD, or LWORD
  bitFields  [7]  IMPLICIT BitFieldDef}
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
BitFieldDef   ::=  SEQUENCE OF {
  bitDef     [2] IMPLICIT OCTET STRING}
                 -- Length is always 2 octets.
                 -- First octet contains starting
                 -- Bit Position.  Trailing octet
                 -- contains the number of bits.
ExpandedStringTypeSpec  ::= UINT-- String Length In Octets
ExpandedString2TypeSpec ::= UINT-- String Length In Octets
AlternateTypeSpec ::=   CHOICE {
  directlyDerivedTypeSpec     [0]   IMPLICIT    TypeID,
  subrangeTypeSpec            [1]   IMPLICIT    SubrangeTypeSpec,
  enumeratedTypeSpec          [2]   IMPLICIT    enumeratedTypeSpec,
  fbTypeSpec                  [3]   IMPLICIT    FBTypeSpec}
SubrangeTypeSpec ::=  SEQUENCE{
  baseType   TypeID,       -- NOTE: minValue and maxValue
  minValue   FixedLengthInteger, -- must be within the range
  maxValue   FixedLengthInteger} -- of baseType values
EnumeratedTypeSpec ::= SEQUENCE OF AnyString
BitNameDefintion ::= SEQUENCE {
  bitName    AnyString,
  bitNumber  USINT}
FBTypeSpec ::= SET{
  inputs          [0] IMPLICIT FbtElementSpec OPTIONAL,
  outputs         [1] IMPLICIT FbtElementTypeSpec OPTIONAL,
  controlInputs   [2] IMPLICIT FbtElementTypeSpec OPTIONAL,
  controlOutputs  [3] IMPLICIT FbtElementTypeSpec OPTIONAL}
FbtElementTypeSpec ::= SEQUENCE OF ElementSpec
ElementSpec ::= SEQUENCE {
  name       AnyString,
  typespec   ElementTypeSpec}
ElementTypeSpec ::= CHOICE {
  [0] IMPLICIT TypeID,
  [1] IMPLICIT SubrangeTypeSpec,
  [2] IMPLICIT EnumeratedTypeSpec,
  [3] IMPLICIT FormalArrayTypeSpec,
  [4] IMPLICIT ExpandedStringTypeSpec,
  [5] IMPLICIT ExpandedString2TypeSpec}
```

Additional elements of “FBTypeSpec” are being considered.

The following END statement terminates the ASN.1 module opened in section C-4.

```
END.
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# C-5 CIP Application Transfer Syntax: Compact Encoding

This section describes the means by which the data types defined in section C-2.1 Data Type Values are encoded/transferred across CIP. The abstract syntax definition along with a particular set of encoding rules results in a transfer syntax. For CIP application user data, a single set of encoding rules is defined (Compact Encoding), resulting in the Compact transfer syntax.

Compact Encoding rules start with the encoding rules defined in ASN.1 8825. Compact Encoding then applies optimization rules, starting with the outer most Service Data Unit (SDU) and progressing to each successive encapsulated SDU. Compact Encoding defines a more efficient encoding mechanism by reducing the amount of information (overhead) transferred between devices.

The difference between a Compact encoded value and an ASN.1 encoded value is the removal of the fields describing the type and length of the information. In other words, the TAG and LENGTH components of an ASN.1-encoded value are not transmitted on CIP. In addition, the Compact Encoding rules indicate that octet-ordering rules are the reverse of those seen in ASN.1.

Given the conditions listed in the next section, the following general rules are applied to an ASN.1 encoded value to generate a Compact encoded value:

- Remove the Identifier Octets. Remove the “TAG” octets specified by ASN.1.
- Remove the Length Octets. Remove the “LENGTH” octets specified by ASN.1.
- Reverse the byte ordering for multiple content octets.

## C-5.1 Compact Encoding Constraints

**Important:** The representation of a variable value using Compact Encoding is only possible with the following restrictions:

- In a multi-peer communication relationship, the entities involved in the pre-established connection have prior knowledge of the variable type, and do not need to transmit the type description with the value of the variable. This knowledge is available by accessing the description of the variable and the variable type.
- The variable type is fixed length and has no conditional or optional fields.
- The encoding of a given variable is represented with a constant number of octets derived from the type specification of this variable.

## C-5.2 Encoding Rules/Examples

This section lists rules specific to the various data types implemented in the CIP system and shows examples of the Compact Encoding.

### C-5.2.1 BOOL Encoding

The Boolean encoding is performed on a single OCTET.

**Table C-5.1 Boolean Encoding**

| If the value is: | Then: |
|---|---|
| FALSE | The octet shall be 0x00 |
| TRUE | The octet shall be 0x01 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The example illustrated below depicts the encoding of a BOOL whose value is FALSE.

**Table C-5.2 Example Compact Encoding of a BOOL Value**

| Octet Number | 1st |
|---|---|
| BOOL | 00 |

### C-5.2.2 SignedInteger Encoding

This section gives you examples of the Compact Encoding of SINT, INT, DINT, LINT data values. A generic illustration of the encoding of SignedInteger values is given below:

**Table C-5.3 SignedInteger Encoding**

| Octet Number | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th | 8th |
|---|---|---|---|---|---|---|---|---|
| SINT | 0LSB | | | | | | | |
| INT | 0LSB | 1LSB | | | | | | |
| DINT | 0LSB | 1LSB | 2LSB | 3LSB | | | | |
| LINT | 0LSB | 1LSB | 2LSB | 3LSB | 4LSB | 5LSB | 6LSB | 7LSB |

The following example illustrates the encoding of a variable of type DINT whose value is 0x12345678.

**Table C-5.4 Example Compact Encoding of a SignedInteger Value**

| Octet Number | 1st | 2nd | 3rd | 4th |
|---|---|---|---|---|
| DINT | 78 | 56 | 34 | 12 |

### C-5.2.3 UnsignedInteger Encoding

This section gives you examples of the Compact Encoding of USINT, UINT, UDINT, ULINT, and ENGUNIT data values. A generic illustration of the encoding of UnsignedInteger values is given below:

**Table C-5.5 UnsignedInteger Encoding**

| Octet Number | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th | 8th |
|---|---|---|---|---|---|---|---|---|
| USINT | 0LSB | | | | | | | |
| UINT | 0LSB | 1LSB | | | | | | |
| UDINT | 0LSB | 1LSB | 2LSB | 3LSB | | | | |
| ULINT | 0LSB | 1LSB | 2LSB | 3LSB | 4LSB | 5LSB | 6LSB | 7LSB |
| ENGUNIT | OLSB | 1LSB | | | | | | |

The example below illustrates the encoding of a variable of type UDINT whose value is 0xAABBCCDD.

**Table C-5.6 Example Compact Encoding of a UnsignedInteger Value**

| Octet Number | 1st | 2nd | 3rd | 4th |
|---|---|---|---|---|
| UDINT | DD | CC | BB | AA |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### C-5.2.4 FixedLengthReal Encoding

This section gives you examples of the Compact Encoding of REAL and LREAL data values. A generic illustration of the encoding of FixedLengthReal values is given below:

**Table C-5.7 FixedLengthReal Encoding**

| Octet Number | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th | 8th |
|---|---|---|---|---|---|---|---|---|
| REAL | 0LSB | 1LSB | 2LSB | 3LSB | | | | |
| LREAL | 0LSB | 1LSB | 2LSB | 3LSB | 4LSB | 5LSB | 6LSB | 7LSB |

The example below illustrates the encoding of a variable (Float1) whose type is REAL and whose value is Float1: = 10.0. (0x41200000 in IEEE format ($+1*1.25*2^3$, sign bit is 0 (+1), exponent is 130(bias 127), fraction is .25)).

**Table C-5.8 Example Compact Encoding of a REAL Value**

| Octet Contents | 0LSB | 1LSB | 2LSB | 3LSB |
|---|---|---|---|---|
| REAL | 00 | 00 | 20 | 41 |

The example below illustrates the encoding of a variable (Float2) whose type is LREAL and whose value is Float2: = –100.0. (0xC059000000000000 in IEEE format ($-1*1.5625*2^6$, sign bit is 1 (-1), exponent is 1029 (bias 1023), fraction is .5625)).

**Table C-5.9 Example Compact Encoding of a LREAL Value**

| Octet Contents | 0LSB | 1LSB | 2LSB | 3LSB | 4LSB | 5LSB | 6LSB | 7LSB |
|---|---|---|---|---|---|---|---|---|
| LREAL | 00 | 00 | 00 | 00 | 00 | 00 | 59 | C0 |

### C-5.2.5 Time Encodings

This section gives you examples of the Compact Encoding of TIME, DATE, TIME_OF_DAY, DATE_AND_TIME, FTIME, LTIME, ITIME data values. A generic illustration of the encoding of Time values is given below:

**Table C-5.10 Time Encoding**

| Octet Number | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th | 8th |
|---|---|---|---|---|---|---|---|---|
| TIME | 0LSB | 1LSB | 2LSB | 3LSB | | | | |
| DATE | 0LSB | 1LSB | | | | | | |
| TIME_OF_DAY | 0LSB | 1LSB | 2LSB | 3LSB | | | | |
| DATE_AND_TIME | 0LSB-Time | 1LSB-Time | 2LSB-Time | 3LSB-Time | 0LSB-Date | 1LSB-Date | | |
| FTIME | 0LSB | 1LSB | 2LSB | 3LSB | | | | |
| LTIME | 0LSB | 1LSB | 2LSB | 3LSB | 4LSB | 5LSB | 6LSB | 7LSB |

### C-5.2.6 String Encodings

This section gives you examples of the Compact Encoding of STRING, STRING2, STRINGN, and SHORT_STRING data values.

**Important:** The preferred string type for user supplied string data is STRING2 due to international character string requirements.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

A generic illustration of the encoding of a STRING value is given below:

**Table C-5.11 STRING Encoding**

| | Contents (charcount) | | Contents (string contents) |
|---|---|---|---|
| STRING | 0LSB | 1LSB | 0LSB |
| | | | 1 byte character |

A generic illustration of the encoding of a STRING2 value is given below:

**Table C-5.12 STRING2 Encoding**

| | Contents (charcount) | | Contents (string2contents) | |
|---|---|---|---|---|
| STRING2 | 0LSB | 1LSB | 0LSB | 1LSB |
| | | | 2 byte character | |

A generic illustration of the encoding of a STRINGN value is given below:

**Table C-5.13 STRINGN Encoding**

| | Contents (charsize) | | Contents (charcount) | | Contents (stringNcontents) | |
|---|---|---|---|---|---|---|
| STRINGN | 0LSB ..... | 1LSB | 0LSB ..... | 1LSB | 0LSB ..... | NLSB |
| | | | | | N byte character | |

A generic illustration of the encoding of a SHORT_STRING value is given below:

**Table C-5.14 SHORT_STRING Encoding**

| | Contents (charcount) | Contents (short_string) |
|---|---|---|
| SHORT_STRING | 0LSB | 0LSB |
| | | 1 byte character |

The example below illustrates the encoding of a string variable whose contents equal ”Mill”. Encoding examples of all string types are presented. Character coding is specified in ISO 10646. The hexadecimal equivalent is: 0x4D696C6C for 8-bit encoding.

The example below encodes ”Mill” as a STRING type.

**Table C-5.15 Example Compact Encoding of A String Value**

| | Contents (charcount) | | Contents (string contents) | | | |
|---|---|---|---|---|---|---|
| STRING | 04 | 00 | 4D | 69 | 6C | 6C |

The example below encodes ”Mill” as a STRING2 type.

**Table C-5.16 Example Compact Encoding of String2 Value**

| | Contents (charcount) | | Contents (string2 contents) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| STRING2 | 04 | 00 | 4D | 00 | 69 | 00 | 6C | 00 | 6C | 00 |

2 byte character


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The example below encodes ”Mill” as a SHORT_STRING type.

**Table C-5.17 Example Compact Encoding of SHORT_STRING Value**

| | Contents (charcount) | Contents (short_string contents) | | | |
|---|---|---|---|---|---|
| SHORT_STRING | 04 | 4D | 69 | 6C | 6C |

## C-5.2.7 FixedLengthBitString Encoding

This section supplies examples of the Compact Encoding of BYTE, WORD, DWORD, LWORD data values. The table below illustrates the bit placement rules associated with the Compact Encoding of a FixedLengthBitString.

**Table C-5.18 Example Compact Encoding of a FixedLengthBitString Value**

| Octet Number→ | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th | 8th |
|---|---|---|---|---|---|---|---|---|
| BYTE | 7.....0 | | | | | | | |
| WORD | 7.....0 | 15.....8 | | | | | | |
| DWORD | 7.....0 | 15.....8 | 23.....16 | 31.....24 | | | | |
| LWORD | 7.....0 | 15.....8 | 23.....16 | 31.....24 | 39.....32 | 47.....40 | 55.....48 | 63.....56 |

The examples below illustrate the encoding of a BYTE, WORD, DWORD, and an LWORD.

**Figure C-5.1 Example Compact Encoding of A BYTE FixedLengthBitString**

```
Bits In Memory:
   7……0
   00001111
CompactEncoded     BYTE
   00001111 or 0x0F
```

**Figure C-5.2 Example Compact Encoding of A WORD FixedLengthBitString**

```
Bits In Memory:
   15……...….0
   00001111  11001111
CompactEncoded     WORD
   11001111  00001111 or 0xCF0F
```

**Figure C-5.3 Example Compact Encoding of A DWORD FixedLengthBitString**

```
Bits In Memory:
   31……...…….…....0
   00001111  11001111  10110110  00111110
CompactEncoded     DWORD
   00111110  10110110  11001111  00001111 or 0x3EB6CF0F
```

**Figure C-5.4 Example Compact Encoding of A LWORD FixedLengthBitString**

```
Bits In Memory:
   63……...…….....0
   11110000  11001111  10110110  00111110  11110000  00101101   00011110  00001111
CompactEncoded     LWORD
   00001111  00011110  00101101  11110000  00111110  10110110  11001111  11110000 or 0x0F1E2DF03EB6CFF0
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### C-5.2.8 Array Encoding

The array encoding uses the encoding rules for the CIP Data types for each element and concatenates the elements which compose the array. The encoded values of the array elements are encoded in the same order as they are declared in the corresponding type or variable specification. These elements may be of any CIP Data type.

The definition of a single–dimensional array in a CIP network is:

```
ARRAY ::= SEQUENCE OF { array_elements, CIPData }
```

Assume the following array definition:

```
ARRAY ::= SEQUENCE OF { array_elements:= 2, UINT }
```

Plugging the UINT values {1,2} into this array definition yields the following encoding:

**Table C-5.19 Example Compact Encoding of a Single Dimensional ARRAY**

| Octet Number | 1st | 2nd | 3rd | 4th |
|---|---|---|---|---|
| ARRAY | 01 | 00 | 02 | 00 |

The definition of a two–dimensional array in a CIP network is:

```
ARRAY ::=   SEQUENCE OF { array_elements,
              SEQUENCE OF { array_elements, CIPData } }
```

The definition of a three–dimensional array in a CIP network is:

```
ARRAY ::=   SEQUENCE OF { array_elements,
              SEQUENCE OF { array_elements,
              SEQUENCE OF { array_elements, CIPData } } }
```

Since CIPData may comprise either ElementaryData or DerivedData, a new type or variable specification may be required before transmitting the values for the ARRAY.

A multi–dimensional array is seen on the wire as a single–dimensional array. The order of the array elements is maintained via the packing/unpacking sequence followed by the end nodes. The sequence followed is to access the Nth dimension of the array for all values of the other dimensions.

This is achieved by first accessing the array with all dimensions set to their initial index values. After this the Nth dimension is incremented through all of its index values. When the end of the index range for the Nth dimension is reached, the (N-1)th dimension is incremented, and the Nth dimension is set to its initial index value. This process is repeated until all of the array's dimensions have reached the ends of their index ranges, and results in the array being packed into the message buffer as a single–dimensional array. The same procedure is followed to unpack the single–dimensional array into a multi–dimensional array.

The two–dimensional array shown results in the data stream below when it is packed into a single–dimensional array following the compact encoding rules:

```
ARRAY1 [2,3] of UINT := { { 1, 2, 3 },{ 4, 5, 6 } }
```

**Table C-5.20 Example Compact Encoding of a Multi-Dimensional ARRAY**

| Octet Number | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th | 8th | 9th | 10th | 11th | 12th |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ARRAY | 01 | 00 | 02 | 00 | 03 | 00 | 04 | 00 | 05 | 00 | 06 | 00 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

### C-5.2.9 Structure Encoding

The structure encoding uses the encoding rules for the CIPData types for each element and concatenates the elements which compose the structure.

The encoded values of the structure elements are encoded in the same order as they are declared in the corresponding ASN.1 type or variable specification. These elements may be of any CIPData type.

STRUCT ::= SEQUENCE OF CIPData –- May be different types

Since CIPData may be comprised of either ElementaryData or DerivedData, a new type or variable specification may be required before transmitting the values for the STRUCT.

Assume the following structure definition:

newStruct ::= SEQUENCE { BOOL,UINT,DINT}

Plugging the values {TRUE, 0x1234, 0x56789ABC} into the structure results in the following encoding:

**Table C-5.21 Example Compact Encoding of a STRUCTURE**

| Octet Number | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th |
|---|---|---|---|---|---|---|---|
| newStruct | 01 | 34 | 12 | BC | 9A | 78 | 56 |

### C-5.2.10 Examples of How More Complex Data Formats are Encoded

The examples below show how more complex data formats are packed. Example 1 shows the packing of an array of structures. Example 2 shows how a structure with an array element is packed.

Example 1: Encoding an Array Of Structures:

```
STRUCT1 ::= SEQUENCE OF {
   UINT     ele1;
   USINT    ele2;
   USINT    ele3;
   USINT    ele4;
   UINT     ele5;
}
ARRAY1 [ 2,3 ] of STRUCT1 := {
   { { 1, 2, 3, 4, 5 }, { 6, 7, 8, 9, 10 },
     { 11, 12, 13, 14, 15 } },
   { { 15, 14, 13, 12, 11 }, { 10, 9, 8, 7, 6 },
     { 5, 4, 3, 2, 1 } } }
```

results in the following data stream:

```
[01][00][02][03][04][05][00] [06][00][07][08][09][0A][00]
[0B][00][0C][0D][0E][0F][00] [0F][00][0E][0D][0C][0B][00]
[0A][00][09][08][07][06][00] [05][00][04][03][02][01][00]
```

Example 2: Encoding a Structure with an Array Element

```
STRUCT2 ::= SEQUENCE OF {
   UINT     ele1;
   ARRAY [ 3 ] of USINT array2;
   UINT     ele5;
}
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
STRUCT2 := { 1, { 2, 3, 4 }, 5 }
```

results in the following data stream:

```
[01][00] [02][03][04] [05][00]
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## C-6 Data Type Reporting

Objects may choose to implement a mechanism for reporting Data Type of a particular Attribute or transmitting type information along with the actual data. This section defines the means by which Data Typing information is conveyed.

The specification of CIP Data Type information utilizes the ASN.1 methodology specified in ISO 8824:1987(E) and ISO 8825:1987(E) with CIP defined optimizations to encode the **DataTypeSpecification** production defined in section C-4.2. The CIP defined optimizations are listed below:

- The Length Octet of a NULL type is not encoded. For example; the encoding of *'abc [PRIVATE 1] IMPLICIT NULL'* would be: $C1_{hex}$ (a tag with no length octet).

The sections that follow detail:

- Elementary Data Type Reporting
- Constructed Data Type Reporting


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## C-6.1 Elementary Data Type Reporting

Elementary data types are identified using the identification codes defined in the table below. These codes illustrate the encoding of the primitive members of the **DataTypeSpecification** production. Remember that CIP specifies that ASN.1 NULL types do not report the Length Octet of zero (0).

**Table C-6.1 Identification Codes and Descriptions of Elementary Data Types**

| Data Type Name | Data Type Code (in hex) | Data Type Description |
|---|---|---|
| BOOL | C1 | Logical Boolean with values TRUE and FALSE |
| SINT | C2 | Signed 8–bit integer value |
| INT | C3 | Signed 16–bit integer value |
| DINT | C4 | Signed 32–bit integer value |
| LINT | C5 | Signed 64–bit integer value |
| USINT | C6 | Unsigned 8–bit integer value |
| UINT | C7 | Unsigned 16–bit integer value |
| UDINT | C8 | Unsigned 32–bit integer value |
| ULINT | C9 | Unsigned 64–bit integer value |
| REAL | CA | 32–bit floating point value |
| LREAL | CB | 64–bit floating point value |
| STIME | CC | Synchronous time information |
| DATE | CD | Date information |
| TIME_OF_DAY | CE | Time of day |
| DATE_AND_TIME | CF | Date and time of day |
| STRING | D0 | character string (1 byte per character) |
| BYTE | D1 | bit string - 8-bits |
| WORD | D2 | bit string - 16-bits |
| DWORD | D3 | bit string - 32-bits |
| LWORD | D4 | bit string - 64-bits |
| STRING2 | D5 | character string (2 bytes per character) |
| FTIME | D6 | Duration (high resolution) |
| LTIME | D7 | Duration (long) |
| ITIME | D8 | Duration (short) |
| STRINGN | D9 | character string (N bytes per character) |
| SHORT_STRING | DA | character sting (1 byte per character, 1 byte length indicator) |
| TIME | DB | Duration (milliseconds) |
| EPATH | DC | CIP path segments |
| ENGUNIT | DD | Engineering Units |
| STRINGI | DE | International Character String |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## C-6.2 Constructed Data Type Reporting

This section details the means by which the structure and array information presented within the DataTypeSpecification production is represented.

**Table C-6.2 Identification Codes and Descriptions of Constructed Data Types**

| Data Type Name | Data Type Code (in hex) | Data Type Description |
|---|---|---|
| Abbreviated Structure | A0 | Structure identified by 16-bit CRC value |
| Abbreviated Array | A1 | Array identified by 16-bit CRC value |
| Formal Structure | A2 | Structure defined by member type list |
| Formal Array | A3 | Array defined by type and size |
| | A4-A7 | Reserved – do not use |
| Formal Handle Structure | A8 | Structure defined by member type and member handle list |
| | A9-AF | Reserved |
| Defined Constructed Data Types | B0 | Defined constructed data types. See C-6.2.3. |
| | B1-BF | Reserved |

### C-6.2.1 Structure Type Definition

CIP defines three different methods for reporting Structure Type Definitions:

- Formal Encoding (FormalStrucTypeSpec)
- Formal Handle Encoding (FormalHandleStrucTypeSpec)
- Abbreviated Encoding (AbbreviatedStrucTypeSpec)

Formal encoding is used to provide a detailed report of the complete structure definition, including the complete definition of all component data types (the structure member number can be obtained based on the component position). Formal Handle encoding is used to provide a detailed report of the complete structure definition, including the complete definition of all component data types and component handles. Abbreviated encoding is used to specify a *shorter* form of the structure definition. This *shorter* form does not include the data types associated with the structure's components.

Formal Encoding for Structure Type Information

Table C-6.3 defines the formal encoding format for non-nested structures.

**Table C-6.3 Formal Structure Encoding Definition**

| Byte | Definition |
|---|---|
| 0 | Formal Structure (A2) |
| 1 | Length of structure definition in bytes (n) |
| 2 – (n+1) | Data type code (C0 – DF) |

The two examples below illustrate formal encoding for structure type specifications. This is actually an example of the encoding of the **FormalStrucTypeSpec** production defined in section C-4.2, Data Type Specification/Dictionaries.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Example 1:**
Table C-6.4 shows the encoding of the following structure definition

```
STRUCT ::= SEQUENCE OF { BOOL, UINT, DINT }
```

**Table C-6.4 Example 1 of Formal Encoding of a Structure Type Specification**

| Struc Type | Type Length | Data Type (BOOL) | Data Type (UINT) | Data Type (DINT) |
|---|---|---|---|---|
| A2 | 03 | C1 | C7 | C4 |

Note that the IMPLICIT NULL types from the **DataTypeSpecification** production are not followed by a Length Octet of zero (0).

**Example 2:**

Table C-6.5 shows the encoding of the following structure definition

```
STRUCT_MAIN ::= SEQUENCE OF { UINT, STRUCT_SUB, INT }
```

with subelement STRUCT_SUB defined as:

```
STRUCT_SUB ::= SEQUENCE OF { UINT, SINT, INT }
```

**Table C-6.5 Example 2 of Formal Encoding of a Structure Type Specification**

| Struct | Type Length | Component | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | Nested Component | | | |
| | | Data Type (UINT) | Struct Type | Type Length | Data Type (UINT) | Data Type (SINT) | Data Type (INT) | Data Type (INT) |
| A2 | 07 | C7 | A2 | 03 | C7 | C2 | C3 | C3 |

Formal Encoding for Handle Structure Type Information

Table C-6.6 defines the formal encoding format for structures with handles.

**Table C-6.6 Formal Structure with Handles Encoding Definition**

| Byte | Definition |
|---|---|
| 0 | Formal Handle Structure (A8) |
| 1 | Length of structure definition in bytes (n) |
| 2 | Handle length (1, 2 or 4 bytes) |
| 3 | Data type code (C0 – DF) [1] |
| 4 thru<br>4 + (Handle length – 1) | Handle (1, 2 or 4 byte value assigned by application) [1] |

Table Footnotes
1 Additional members are represented by pairs of data type codes and handles

The two examples below illustrate formal encoding for handle structure type specifications. This is actually an example of the encoding of the **FormalHandleStrucTypeSpec** production defined in section C-4.2, Data Type Specification/Dictionaries.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Example 3:**

Table C-6.7 shows the encoding of the following structure definition

```
STRUCT ::= SEQUENCE OF { BOOL, UINT, DINT }
```

**Table C-6.7 Example 3 of Formal Encoding of a Handle Structure Type Specification**

| Struc Type | Type Length | Handle Length | Data Type (BOOL) | Member Handle | Data Type (UINT) | Member Handle | Data Type (DINT) | Member Handle |
|---|---|---|---|---|---|---|---|---|
| A8 | 10 | 04 | C1 | D9 B9 74 3E | C7 | F4 15 59 93 | C4 | 0B 7B 24 A6 |

Example 4:

Table C-6.8 shows the encoding of the following structure definition

```
STRUCT_MAIN ::= SEQUENCE OF { UINT, STRUCT_SUB, INT }
```

with sub element STRUCT_SUB defined as:

```
STRUCT_SUB ::= SEQUENCE OF { SINT, INT }
```

**Table C-6.8 Example 4 of Formal Encoding of a Handle Structure Type Specification**

| Struc Type | Type Length | Handle Length | Component | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Data Type (UINT) | Member Handle | Struct Type | Type Length | Handle Length | Nested Component | | | | Data Type (INT) | Handle |
| | | | | | | | | Data Type (SINT) | Member Handle | Data Type (INT) | Member Handle | | |
| A8 | 0E | 02 | C7 | 90 1D | A8 | 05 | 01 | C2 | F8 | C3 | B2 | C3 | 36 ED |

Abbreviated Encoding for Structure Type Information

Table C-6.9 defines the abbreviated encoding format for structures.

**Table C-6.9 Abbreviated Structure Encoding Definition**

| Byte | Definition |
|---|---|
| 0 | Abbreviated Structure (A0) |
| 1 | Length of abbreviated structure (02) |
| 2 – 3 | UINT containing CRC |

The example below illustrates the abbreviated encoding for structure type specifications. This is actually an example of the encoding of the **AbbreviatedStrucTypeSpec** production defined in section C-4.2, Data Type Specification/Dictionaries.

The UINT defined within the AbbreviatedStrucTypeSpec production is initialized with a 16 bit Cyclic Redundancy Check (CRC) value. This can be used by application logic to determine whether or not the format of the structure has changed. The byte stream used to produce the CRC is the actual formally encoded (**FormalStrucTypeSpec)** structure type specification. See Section C-7, CRC Generation Algorithms. Note that the algorithms presented in section C-7 were used to generate the CRC value of 0x5159 from the Formally Encoded structure type specification [A2][03][C7][C2][C3].

**Example 5:**
Table C-6.10 shows the abbreviated encoding of the structure definition presented in the previous section:

**Table C-6.10 Example of Abbreviated Encoding of a Structure Type Specification**

| Struc | Type Length | UINT Containing CRC | |
|---|---|---|---|
| A0 | 02 | C7 | 26 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

Note that the algorithms presented in section C-7 were used to generate the CRC value of $26C7_{hex}$ from the Formally Encoded type specification:
[A2][07][C7][A2][03][C7][C2][C3][C3].

### C-6.2.2 Array Type Definition

CIP defines two different methods for reporting Array Type Definitions:

- Formal Encoding (FormalArrayTypeSpec)
- Abbreviated Encoding (AbbreviatedArrayTypeSpec)

Formal encoding is used to provide a detailed report of the complete array definition, including the data content and the array's dimensions. Abbreviated encoding is used to specify a *shorter* form of the array definition. This *shorter* form does not include information specifying the array's dimensions.

Formal Encoding for Array Type Information

Table C-6.11 defines the formal encoding format for arrays.

**Table C-6.11 Formal Array Encoding Definition**

| Byte | Definition |
|---|---|
| 0 | Formal Array (A3) |
| 1 thru 4 | Number of elements in the dimension (UDINT) |
| 5 thru end | One of:<br>1 Array element data type code (C0 – DF)<br>2 Formal array encoding[1]<br>3 Formal structure encoding<br>4 Formal handle structure encoding |
| Table Footnotes:<br>1. Each dimension of the array must be represented by a formal array | |

The two examples below illustrate formal encoding for structure type specifications. This is actually an example of the encoding of the **FormalArrayTypeSpec** production defined in section C-4.2, Data Type Specification/Dictionaries.

**Example 1:**
This is an example of a 1 dimensional array (10 elements) of UINT.

Table C-6.12 shows the formal encoding of the following array definition.

ARRAY1 ::= SEQUENCE OF { array_elements := 10, UINT }

**Table C-6.12 Example 1 of Formal Encoding of An Array Type Specification**

| Array | ArrayElements (byte 0) | ArrayElements (byte 1) | ArrayElements (byte 2) | ArrayElements (byte 3) | Data Type (UINT) |
|---|---|---|---|---|---|
| A3 | 0A | 00 | 00 | 00 | C7 |

Note that the IMPLICIT NULL types from the **DataTypeSpecification** production are not followed by a Length Octet of zero (0).

**Example 2:**
This is an example of a 2 dimensional array (20 elements by 256 elements) of a structure consisting of a UINT, SINT and INT.

Figure C-6.1 shows the encoding of the following array definition


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
ARRAY1 ::= SEQUENCE OF {array_elements := 20,
              SEQUENCE OF {array_elements := 256, STRUCT_ELE}
              }
```

in which STRUCT_ELE is defined as:

```
STRUCT_ELE ::= SEQUENCE OF { UINT, SINT, INT }
```

**Figure C-6.1 Example 2 of Formal Encoding of an Array Type Specification**

Formal Encoding:

[A3][14][00][00][00][A3][00][01][00][00][A2][03][C7][C2][C3]

| Array Type | Array Elements (byte 0) | Array Elements (byte 1) | Array Elements (byte 2) | Array Elements (byte 3) |
|---|---|---|---|---|
| A3 | 14 | 00 | 00 | 00 |

| | Array Contents | | | | |
|---|---|---|---|---|---|
| | Array Type | Array Elements (byte 0) | Array Elements (byte 1) | Array Elements (byte 2) | Array Elements (byte 3) |
| … | A3 | 00 | 01 | 00 | 00 |

| | Nested Array Contents | | | | |
|---|---|---|---|---|---|
| | Struct Type | Type Length | UINT | SINT | INT |
| … | A2 | 03 | C7 | C2 | C3 |

Note that the IMPLICIT NULL types from the **DataTypeSpecification** production are not followed by a Length Octet of zero (0).

Abbreviated Tag Encoding for Array Type Information

Table C-6.13 defines the abbreviated encoding format for arrays.

**Table C-6.13 Abbreviated Array Encoding Definition**

| Byte | Definition |
|---|---|
| 0 | Abbreviated Array (A1) |
| 1 | Length of abbreviated array (n) |
| 2 thru (2+n-1) | One of:<br>1 Array element data type code (C0 – DF)<br>2 Abbreviated array encoding [1]<br>3 Abbreviated structure encoding |

Table Footnotes:

1 Each dimension of the array must be represented by an abbreviated array

The abbreviated encoding of an Array type DOES NOT include the information specifying the Array's dimensions. This is actually an example of the encoding of the **AbbreviatedArrayTypeSpec** production defined in section C-4.2, Data Type Specification/Dictionaries.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

**Example 1:**

This is an example of a 1 dimensional array (10 elements) of UINT.

Table C-6.14 shows the abbreviated encoding of the following array definition:

```
ARRAY2 ::= SEQUENCE OF {array_elements := 10, UINT}
```

**Table C-6.14 Example 1 of Abbreviated Encoding of an Array Type Specification**

| Array Type | Type Length | Data Type (UINT ) |
|---|---|---|
| A1 | 01 | C7 |

Note that the IMPLICIT NULL types from the **DataTypeSpecification** production are not followed by a Length Octet of zero (0).

**Example 2:**

This is an example of a 2 dimensional array (20 elements by 900 elements) of a structure consisting of a UINT, SINT and INT.

Table C-6.15 shows the abbreviated encoding of the following array definition:

```
ARRAY ::= SEQUENCE OF {array_elements := 20,
            SEQUENCE OF {array_elements := 900, STRUCT_ELE}
             }
```

in which STRUCT_ELE is defined as:

```
STRUCT_ELE ::= SEQUENCE OF { UINT, SINT, INT }
```

**Table C-6.15 Example 2 of Abbreviated Encoding of an Array Type Specification**

| Array Type | Type Length | Array Type | Type Length | Structure (STRUCT_ELE) | | | |
|---|---|---|---|---|---|---|---|
| | | | | Struct Type | Type Length | UINT Containing CRC | |
| A1 | 06 | A1 | 04 | A0 | 02 | 59 | 51 |

Note that the algorithms presented in section C-7 were used to generate the CRC value of 0x5159 from the Formally Encoded structure type specification: [A2][03][C7][C2][C3].

### C-6.2.3 Defined Constructed Data Types

Defined Constructed Data Types are data types whose structure is predefined and usage is common throughout the CIP Networks Library.

**Table C-6.16 Defined Constructed Data Types**

| Defined Constructed Data Type Code (UINT) | Keyword | Description |
|---|---|---|
| 0 | | Reserved |
| 1 | SIGNED_ODOMETER | STRUCT of $x10^{n}$, $x10^{n+3}$, $x10^{n+6}$, $x10^{n+9}$, $x10^{n+12}$<br>Signed large magnitude counter. See section C-2.1.5.1 |
| 2 | ODOMETER | STRUCT of $x10^{n}$, $x10^{n+3}$, $x10^{n+6}$, $x10^{n+9}$, $x10^{n+12}$<br>Unsigned large magnitude counter. See section C-2.1.5.2 |
| 3 | ETH_MAC_ADDR | Ethernet MAC Address. See section C-2.1.5.3 |
| 4-65535 | | Reserved for future use |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The formal encoding for any Defined Constructed Data Type is B0 followed by the associated Common Constructed Data Type Code from Table C-6.17 encoded as a UINT.

Table C-6.17 Defines the formal encoding format for Defined Constructed Data Types.

**Table C-6.17 Defined Constructed Data Type Encoding Definition**

| Byte | Definition |
|---|---|
| 0 | Defined Constructed Data Type (B0) |
| 1 | Type Code (low byte) |
| 2 | Type Code (high byte) |

**Table C-6.18 Defined Constructed Data Type Formal Encoding Example**

| Data Type Code | Defined Constructed Data Type Code (SIGNED_ODOMETER) |
|---|---|
| B0 | 01 00 |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## C-7 CRC Generation Algorithms

The *C* routine below generates a CRC via the polynomial driven method.

```
/*****************************************************************
Function Name:  calcPolyCrc()
Description:  calculates a Cyclic Redundancy Checksum CRC) via the
polynomial driven method.
Inputs:   start_addr - the address of the first byte which is involved
in the crc calculation
     size - the number of consecutive bytes that are involved in the
crc calculation. Size of 0 is invalid.
Outputs:  a 16 bit value containing the calculated CRC
*****************************************************************/
unsigned16BitInteger calcPolyCrc(start_addr, size)
unsigned8BitInteger *start_addr;
unsigned32BitInteger size;
{
unsigned16BitInteger crc;
unsigned8BitInteger  carry;
unsigned8BitInteger  b;
unsigned32BitInteger cnt;
crc = 0;
while (size)
    {
    b = (unsigned8BitInteger) *start_addr++;
    crc ^= b;
    cnt = 0;
    while (cnt < 8)
        {
        carry = crc & 1;
        crc >>= 1;
        if (carry)
           {
           crc ^= 0xa001;
           cnt++;
           }
        }
    size--;
    }
    return (crc);
}
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

The *C* routine below generates a CRC via the table driven method.

```
unsigned16BitInteger poly_table[] = {0x0000,    0xC0C1,     0xC181,
0x0140,     0xC301,     0x03C0,     0x0280,     0xC241,     0xC601,
0x06C0,     0x0780,     0xC741,     0x0500,     0xC5C1,     0xC481,
0x0440,     0xCC01,     0x0CC0,     0x0D80,     0xCD41,     0x0F00,
0xCFC1,     0xCE81,     0x0E40,     0x0A00,     0xCAC1,     0xCB81,
0x0B40,     0xC901,     0x09C0,     0x0880,     0xC841,     0xD801,
0x18C0,     0x1980,     0xD941,     0x1B00,     0xDBC1,     0xDA81,
0x1A40,     0x1E00,     0xDEC1,     0xDF81,     0x1F40,     0xDD01,
0x1DC0,     0x1C80,     0xDC41,     0x1400,     0xD4C1,     0xD581,
0x1540,     0xD701,     0x17C0,     0x1680,     0xD641,     0xD201,
0x12C0,     0x1380,     0xD341,     0x1100,     0xD1C1,     0xD081,
0x1040,     0xF001,     0x30C0,     0x3180,     0xF141,     0x3300,
0xF3C1,     0xF281,     0x3240,     0x3600,     0xF6C1,     0xF781,
0x3740,     0xF501,     0x35C0,     0x3480,     0xF441,     0x3C00,
0xFCC1,     0xFD81,     0x3D40,     0xFF01,     0x3FC0,     0x3E80,
0xFE41,     0xFA01,     0x3AC0,     0x3B80,     0xFB41,     0x3900,
0xF9C1,     0xF881,     0x3840,     0x2800,     0xE8C1,     0xE981,
0x2940,     0xEB01,     0x2BC0,     0x2A80,     0xEA41,     0xEE01,
0x2EC0,     0x2F80,     0xEF41,     0x2D00,     0xEDC1,     0xEC81,
0x2C40,     0xE401,     0x24C0,     0x2580,     0xE541,     0x2700,
0xE7C1,     0xE681,     0x2640,     0x2200,     0xE2C1,     0xE381,
0x2340,     0xE101,     0x21C0,     0x2080,     0xE041,     0xA001,
0x60C0,     0x6180,     0xA141,     0x6300,     0xA3C1,     0xA281,
0x6240,     0x6600,     0xA6C1,     0xA781,     0x6740,     0xA501,
0x65C0,
0x6480,     0xA441,     0x6C00,     0xACC1,     0xAD81,     0x6D40,
0xAF01,     0x6FC0,     0x6E80,     0xAE41,     0xAA01,     0x6AC0,
0x6B80,     0xAB41,     0x6900,     0xA9C1,     0xA881,     0x6840,
0x7800,     0xB8C1,     0xB981,     0x7940,     0xBB01,     0x7BC0,
0x7A80,     0xBA41,     0xBE01,     0x7EC0,     0x7F80,     0xBF41,
0x7D00,     0xBDC1,     0xBC81,     0x7C40,     0xB401,     0x74C0,
0x7580,     0xB541,     0x7700,     0xB7C1,     0xB681,     0x7640,
0x7200,     0xB2C1,     0xB381,     0x7340,     0xB101,     0x71C0,
0x7080,     0xB041,     0x5000,     0x90C1,     0x9181,     0x5140,
0x9301,     0x53C0,     0x5280,     0x9241,     0x9601,     0x56C0,
0x5780,     0x9741,     0x5500,     0x95C1,     0x9481,     0x5440,
0x9C01,     0x5CC0,     0x5D80,     0x9D41,     0x5F00,     0x9FC1,
0x9E81,     0x5E40,     0x5A00,     0x9AC1,     0x9B81,     0x5B40,
0x9901,     0x59C0,     0x5880,     0x9841,     0x8801,     0x48C0,
0x4980,     0x8941,     0x4B00,     0x8BC1,     0x8A81,     0x4A40,
0x4E00,     0x8EC1,     0x8F81,     0x4F40,     0x8D01,     0x4DC0,
0x4C80,     0x8C41,     0x4400,     0x84C1,     0x8581,     0x4540,
0x8701,     0x47C0,     0x4680,     0x8641,     0x8201,     0x42C0,
0x4380,     0x8341,     0x4100,     0x81C1,     0x8081,
0x4040};
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

```
/*************************************************************
FFunction Name:   calcTableCrc()
Description:      Calculates a 16 bit crc value based on the input
                  parameters via the table driven method
Inputs:           start_addr - pointer to the first byte of memory
                  involved in the calculation of the CRC
                  size - the number of bytes to include in the CRC
                  generation
Outputs:          a 16 bit CRC
*************************************************************/
unsigned16BitInteger calcTableCrc(start_addr, size)
unsigned8BitInteger *start_addr;
unsigned32BitInteger size;
{
unsigned16BitInteger crc;
unsigned16BitInteger data;

crc = 0;
while (size)
    {
    data = (unsigned16BitInteger) * start_addr++;
    crc = ((crc >> 8) ^ poly_table[(crc ^ data) & 255]);
    size--;
    }
    return (crc);
}
```


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This page is intentionally left blank


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## Volume 1: Common Industrial Protocol (CIP™) Specification

# Appendix D: Engineering Units


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# Contents

D-1 Introduction ..... 3
D-1.1 Reporting Engineering Units (ENGUNIT Codes) ..... 3
D-1.2 Groups of Engineering Units ..... 4
D-1.3 Naming Conventions ..... 5
D-2 Tables ..... 7
D-2.1 Profile Specific Engineering Unit Enumerations ..... 7
D-2.2 Vendor Specific Engineering Unit Enumerations ..... 7
D-2.3 General ..... 7
D-2.4 Time (includes Date) ..... 8
D-2.5 Temperature ..... 8
D-2.6 Pressure ..... 9
D-2.7 Flow ..... 9
D-2.8 Acceleration ..... 13
D-2.9 Amount of Substance ..... 13
D-2.10 Angle ..... 13
D-2.11 Area ..... 13
D-2.12 Capacitance ..... 14
D-2.13 Charge (Electric Charge) ..... 14
D-2.14 Conductance (Electric Conductance) ..... 14
D-2.15 Current (Electric Current) ..... 15
D-2.16 Energy (Heat, Work) ..... 15
D-2.17 Force ..... 16
D-2.18 Frequency (includes RPM) # ..... 16
D-2.19 Inductance ..... 18
D-2.20 Inertia ..... 18
D-2.21 Length (Distance, Displacement) ..... 18
D-2.22 Light (Luminous Intensity) ..... 19
D-2.23 Magnetic Flux ..... 19
D-2.24 Mass ..... 19
D-2.25 Power (Wattage) ..... 20
D-2.26 Radiology ..... 20
D-2.27 Resistance (Electric Resistance) ..... 21
D-2.28 Sound ..... 21
D-2.29 Torque (Moment of Force) ..... 22
D-2.30 Velocity (Speed) # ..... 22
D-2.31 Dynamic Viscosity ..... 22
D-2.32 Voltage ..... 23
D-2.33 Volume ..... 24
D-2.34 Density ..... 25
D-2.35 Relative Density ..... 26
D-2.36 Electric Conductivity ..... 26
D-2.37 Analytics ..... 26
D-2.38 Kinematic Viscosity# ..... 27


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# D-1 Introduction

In the physical sciences, quantities are measured by comparing them to a quantity whose magnitude is defined to be unity. Such a quantity is called a *unit*. For example, a commonly used unit for mass is the kilogram.

Within the CIP specification, units are often used to describe a device's interface to the physical world. For example, a photoelectric switch might use units of length to describe its min/max detect distances, and a position controller might use units of velocity and acceleration to describe its movements.

This appendix defines a system of units for use in the CIP specification, including names, symbols, and a mechanism for reporting units. This system of units uses the commonly recognized name *engineering units*.

## D-1.1 Reporting Engineering Units (ENGUNIT Codes)

An attribute of data type ENGUNIT is used to specify the engineering units for one or more other attributes.

By providing a ENGUNIT attribute, an object can manage different engineering units. For example, an analog sensor object with a ENGUNIT attribute could potentially measure temperature, flow, pressure, and so on.

If the engineering unit used by an object does not vary, the unit can be specified in the object's specification, and a ENGUNIT attribute is not needed. In such cases, names and symbols of units listed in the object's specification should remain consistent with this appendix.

The ENGUNIT code is used to represent engineering units only, and does not imply a specific data type, range, or scaling. For example, if the engineering units of a UINT (16-bit unsigned integer) attribute is defined by another ENGUNIT attribute which indicates milliamps, this does **not** imply a range of 0-65535 mA. The encoding of a given unit by an object is defined entirely by that object's specification.

The ENGUNIT data type is encoded as a UINT (16-bit unsigned integer). The most significant byte of this UINT selects a group of similar units, and the least significant byte selects a specific unit within that group.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## D-1.2 Groups of Engineering Units

The following groups of engineering units are listed in the tables of section D-2:

| Value of MSB | Engineering Units Group |
|---|---|
| 00 hex thru 07 hex | Reserved |
| 08 hex thru 0F hex | Vendor specific enumerations |
| 10 hex | General |
| 11 hex | Time (includes Date) |
| 12 hex | Temperature |
| 13 hex | Pressure |
| 14 hex | Flow |
| 15 hex | Acceleration |
| 16 hex | Amount of Substance |
| 17 hex | Angle |
| 18 hex | Area |
| 19 hex | Capacitance |
| 1A hex | Charge (Electric Charge) |
| 1B hex | Conductance (Electric Conductance) |
| 1C hex | Current (Electric Current) |
| 1D hex | Energy (Heat, Work) |
| 1E hex | Force |
| 1F hex | Frequency (includes RPM) |
| 20 hex | Inductance |
| 21 hex | Inertia |
| 22 hex | Length (Distance, Displacement) |
| 23 hex | Light (Luminous Intensity) |
| 24 hex | Magnetic Flux |
| 25 hex | Mass |
| 26 hex | Power (Wattage) |
| 27 hex | Radiology |
| 28 hex | Resistance (Electric Resistance) |
| 29 hex | Sound |
| 2A hex | Torque (Moment of Force) |
| 2B hex | Velocity (Speed) |
| 2C hex | Viscosity |
| 2D hex | Voltage |
| 2E hex | Volume |
| 2F hex | Density |
| 30 hex | Relative Density |
| 31 hex | Electric Conductivity |
| 32 hex | Analytics |
| 33 hex | Kinematic Viscosity |
| 34-FF hex | reserved for future standardization |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## D-1.3 Naming Conventions

In the tables of section D-2, each unit lists:

- *Value*: Value used to represent the unit in a ENGUNIT attribute
- *Name*: Commonly used name for the unit (often used for text or display)
- *Symbol*: Commonly used symbol for the unit (often used for expressions)
- *Base Units*: Expression in terms of other base units (such as metric conversions)

Many of the names, symbols, and expressions are taken from the International System of Units (SI). [#1]

In addition, the following conventions are taken from SI specifications:

- The following SI prefixes are used for names and symbols in order to form decimal multiples and submultiples of units as appropriate:

| Factor | Name Prefix | Symbol Prefix |
|---|---|---|
| $10^{12}$ | tera | T |
| $10^{9}$ | giga | G |
| $10^{6}$ | mega | M |
| $10^{3}$ | kilo | k |
| $10^{2}$ | hecto | h |
| $10^{1}$ | deka (or deca) | da |
| $10^{-1}$ | deci | d |
| $10^{-2}$ | centi | c |
| $10^{-3}$ | milli | m |
| $10^{-6}$ | micro | µ |
| $10^{-9}$ | nano | n |
| $10^{-12}$ | pico | p |
| $10^{-15}$ | femto | f |

- Unit symbols and unit names are not used together. For example, millivolt and mV are valid, but mvolt and milliV are not valid.
- A name (symbol) prefix is inseparable from its unit name (symbol). For example, milli amp and milli-amp are not valid names.
- A centered dot ('•') is used to indicate multiplication, and a solidus ('/') is used to indicate division.
- Each unit definition is listed in its singular form. For example, velocity is listed as meter per second, not meters per second. Plural forms are more commonly used in values for a given unit (such as 8 meters per second).
- Capitalization follows SI conventions.
- Only one pre-decimal point position for unit conversions (nonzero number) shall be given in the "Representation in SI Units" column
- Derived units of two or more base SI units can be used in the "Representation in SI Units" column to reduce complexity (e.g. "Pa" instead of "$m^{-1} \bullet kg \bullet s^{-2}$")
- Derived units shall be defined in base SI units in the "Representation in SI Units" column

[1] The International System of Units (SI), NIST Special Publication 330, United States Department of Commerce, Technology Administration, National Institute of Standards and Technology. See also: Guide for the Use of the International System of Units (SI), NIST Special Publication 811. These documents may be available for access on www.nist.gov.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

- The "Representation in SI Units" column shall always give the conversion from the unit described in the "Symbol" column to the base unit (e.g. 1km=$10^3$m), so in general the format looks like this: "symbol = offset + (factor • SI unit [#2])". If the offset is zero or the factor is one, then the corresponding value should not be shown in this column. (e.g. Kelvin → correct: K; wrong: 0 + 1 • K)
- If the symbol already represents the conversion, it shall still be given in the last column (e.g. 1202hex - Kelvin)
- If a unit column is empty it can mean either a formula which needs additional reference material has to be used or no additional value results from a formula (e.g. 1107 hex – DATE)

Also, in order to offset footnote markers from other superscripts, each footnote marker is preceded by the # symbol.

[2] This implies derived units according to the rules in D-1.3.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

# D-2 Tables

## D-2.1 Profile Specific Engineering Unit Enumerations

| Value | Name | Symbol | Base Units |
|---|---|---|---|
| 0000 hex through 07FF hex | Reserved to prevent overlap with the SEMI Standard | | |

## D-2.2 Vendor Specific Engineering Unit Enumerations

| Value | Name | Symbol | Base Units |
|---|---|---|---|
| 0800 hex through 0FFF hex | This range of values is reserved for Engineering Units specific to a particular vendor's needs. Any device that utilizes values within this range must enumerate (or otherwise identify) the exact meaning of each value that can be represented. These values are reserved for previously un-enumerated Engineering Units. Engineering units that are already represented by enumer-ations in the following sections are not allowed to be assigned in this vendor specific range. | | |

## D-2.3 General

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1000 hex | unspecified | | |
| 1001 hex | counts | | |
| 1002 hex | part per million (concentration) | ppm | |
| 1003 hex | position | | |
| 1004 hex | pixel | | |
| 1005 hex | bit | Bit | |
| 1006 hex | byte | Byte | 8 • bit |
| 1007 hex | percent | % | |
| 1008 hex | part per billion (concentration) | ppb | |
| 1009 hex | kilobyte | kB | $10^3$ • Byte |
| 100A hex | megabyte | MB | $10^6$ • Byte |
| 100B hex | gigabyte | GB | $10^9$ • Byte |
| 100C hex | per centimeter | 1/cm | 100/m |
| 100D hex | per meter | 1/m | 1/m |
| 100E hex | per millimeter | 1/mm | 1000/m |
| 100F hex | per kelvin | 1/K | 1/K |
| 1010-10FF hex | reserved for future standardization | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## D-2.4 Time (includes Date)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1100 hex | second [#3] | s | s |
| 1101 hex | millisecond [#4] | ms | $10^{-3} \bullet s$ |
| 1102 hex | microsecond [#5] | µs | $10^{-6} \bullet s$ |
| 1103 hex | minute | min | 60 • s |
| 1104 hex | hour | h | 3600 • s |
| 1105 hex | day | d | 86400 • s |
| 1106 hex | nanosecond | ns | $10^{-9} \bullet s$ |
| 1107 hex | DATE [#6] | D | |
| 1108 hex | TIME_OF_DAY [#7] | TOD | |
| 1109 hex | DATE_AND_TIME [#8] | DT | |
| 110A hex | kilosecond | ks | $10^{3}$ • s |
| 110B hex | week | week | 604800 • s |
| 110C hex | month | month | |
| 110D-11FF hex | reserved for future standardization | | |

## D-2.5 Temperature

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1200 hex | degree Celsius [#9] | °C | -273.15 + K |
| 1201 hex | degree Fahrenheit | °F | -459.67 + 5/9 • K |
| 1202 hex | kelvin [#10] | K | K |
| 1203 hex | Degree Rankine | °R | 5/9 • K |
| 1204 hex | decidegree Celsius | °C/10 | $-27.315 + 10^{-1}$ • K |
| 1205 hex | decidegree Fahrenheit | °F/10 | $-45.967 + 5/9 \bullet 10^{-1} \bullet K$ |
| 1206 hex | decikelvin | K/10 | $10^{-1} \bullet K$ |
| 1207 hex | decidegree Rankine | °R/10 | $5/9 \bullet 10^{-1} \bullet K$ |
| 1208-12FF hex | reserved for future standardization | | |

[3] The second is an SI base unit, defined as the duration of 9192631770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of the cesium-133 atom.
[4] When the *millisecond* unit is encoded with INT data type, it is equivalent to the ITIME data type defined in Appendix C. When the *millisecond* unit is encoded with DINT data type, it is equivalent to the TIME data type defined in Appendix C.
[5] When the *microsecond* unit is encoded with DINT data type, it is equivalent to the FTIME data type defined in Appendix C. When the *microsecond* unit is encoded with LINT data type, it is equivalent to the LTIME data type defined in Appendix C.
[6] The *DATE* unit and its symbol refer to the CIP DATE data type defined in Appendix C.
[7] The *TIME_OF_DAY* unit and its symbol refer to the CIP TIME_OF_DAY data type defined in Appendix C.
[8] The *DATE_AND_TIME* unit and its symbol refer to the CIP DATE_AND_TIME data type defined in Appendix C.
[9] Although the correct SI name is *degree Celsius*, this unit is often named *degree centigrade*.
[10] The kelvin is an SI base unit, defined as the fraction 1/273.16 of the thermodynamic temperature of the triple point of water.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## D-2.6 Pressure

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1300 hex | pound-force per square inch (psi) | psi | $6.894757 \bullet 10^3 \bullet Pa$ |
| 1301 hex | torr | Torr | $(101325/760) \bullet Pa$ |
| 1302 hex | millitorr | mTorr | $(101325/760) \bullet 10^{-3} \bullet Pa$ |
| 1303 hex | millimeter of mercury (at 0°C) | mmHg (0°C) | $1.33322 \cdot 10^2 \cdot Pa$ |
| 1304 hex | inch of mercury (at 0°C) | inHg (0°C) | $3.38638 \bullet 10^3 \bullet Pa$ |
| 1305 hex | centimeter of water (at 25°C) | $cmH_2O$ (25°C) | |
| 1306 hex | inch of water (at 25°C) | $inH_2O$ (25°C) | |
| 1307 hex | bar | bar | $10^5 \bullet Pa$ |
| 1308 hex | millibar | mbar | $10^2 \bullet Pa$ |
| 1309 hex | pascal | Pa | $m^{-1} \bullet kg \bullet s^{-2}$ |
| 130A hex | kilopascal | kPa | $10^3 \bullet Pa$ |
| 130B hex | standard atmosphere | atm | $101325 \bullet Pa$ |
| 130C hex | gram-force per square centimeter | $gf/cm^3$ | $9.80665 \bullet 10^1 \bullet Pa$ |
| 130D hex | millimeter of water (at 25°C) | $mmH_2O$ (25°C) | |
| 130E hex | hectopascal | hPa | $10^2 \cdot Pa$ |
| 130F hex | millimeter of water (at 68°F) | mmH2O (68°F) | $9.789 \cdot Pa$ |
| 1310 hex | meter of water (at 68°F) | mH2O (68°F) | $9.789 \cdot 10^3 \cdot Pa$ |
| 1311 hex | foot of water (at 68°F) | ftH2O (68°F) | $2.98369 \cdot 10^3 \cdot Pa$ |
| 1312 hex | megapascal | MPa | $10^6 \cdot Pa$ |
| 1313 hex | gram force per square centimeter | $gf/cm^2$ | $9.80665 \cdot 10 \cdot Pa$ |
| 1314 hex | kilogram force per square centimeter | $kgf/cm^2$ | $9.80665 \cdot 10^4 \cdot Pa$ |
| 1315 hex | pound force per square foot | $lbf/ft^2$ | $4.78802 \cdot 10 \cdot Pa$ |
| 1316 hex | gigapascal | Gpa | $10^9 \cdot Pa$ |
| 1317 hex | bar absolute | bar (a) | |
| 1318 hex | millipascal | mPa | $10^{-3} \cdot Pa$ |
| 1319 hex | micropascal | µPa | $10^{-6} \cdot Pa$ |
| 131A-13FF hex | reserved for future standardization | | |

## D-2.7 Flow

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1400 hex | standard cubic centimeter per minute | SCCM | |
| 1401 hex | standard liter per minute | SLM | |
| 1402 hex | cubic foot per minute | CFM | $4.71947 \cdot 10^{-4} \cdot m^3 / s$ |
| 1403 hex | pascal-cubic meter per second | $(Pa \bullet m^3)/s$ | $(Pa \bullet m^3)/s$ |
| 1404 hex | kilogram per second | kg/s | kg/s |
| 1405 hex | cubic meter per second | $m^3/s$ | $m^3/s$ |
| 1406 hex | liter per second | l/s | $10^{-3} \bullet m^3/s$ |
| 1407 hex | milliliter per second | ml/s | $10^{-6} \bullet m^3/s$ |
| 1408 hex | gallon per second (us) | gal/s(us) | $3.785412 \cdot 10^{-3} \cdot m^3/s$ |
| 1409 hex | gallon per minute (us) | gal/min(us) | $6.309196 \cdot 10^{-5} \cdot m^3/s$ |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 140A hex | gallon per hour (us) | gal/hr(us) | $1.051503 \cdot 10^{-6} \cdot m^3/s$ |
| 140B hex | pound per second | lb/s | $4.535924 \cdot 10^{-1} \cdot kg/s$ |
| 140C hex | pound per minute | lb/min | $7.55987283 \cdot 10^{-3} \cdot kg/s$ |
| 140D hex | pound per hour | lb/h | $1.25997881 \cdot 10^{-4} \cdot kg/s$ |
| 140E hex | milligram per minute | mg/min | $1.66667 \cdot 10^{-8} \bullet kg/s$ |
| 140F hex | gram per minute | g/min | $1.666667 \cdot 10^{-5} \cdot kg/s$ |
| 1410 hex | kilogram per hour | kg/h | $1.666667 \cdot 10^{-2} \cdot kg/s$ |
| 1411 hex | milliliter per minute | ml/m | $1.666667 \cdot 10^{-8} \bullet m^3/s$ |
| 1412 hex | milliliter per hour | ml/h | $2.777778 \cdot 10^{-10} \bullet m^3/s$ |
| 1413 hex | liter per minute | l/m | $1.666667 \cdot 10^{-5} \bullet m^3/s$ |
| 1414 hex | liter per hour | l/h | $2.777778 \cdot 10^{-7} \bullet m^3/s$ |
| 1415 hex | acre foot per day | af/d | $1.42764 \cdot 10^{-2} \cdot m^3/s$ |
| 1416 hex | acre foot per hour | af/h | $3.42634 \cdot 10^{-1} \cdot m^3/s$ |
| 1417 hex | acre foot per minute | af/min | $2.05580 \cdot 10^{1} \cdot m^3/s$ |
| 1418 hex | acre foot per second | af/s | $1.23348 \cdot 10^{3} \cdot m^3/s$ |
| 1419 hex | barrel per day (us; beer) | bbl/d(us;beer) | $1.358192 \cdot 10^{-6} \cdot m^3/s$ |
| 141A hex | barrel per hour (us; beer) | bbl/h(us;beer) | $3.259660 \cdot 10^{-5} \cdot m^3/s$ |
| 141B hex | barrel per minute (us; beer) | bbl/min(us;beer) | $1.955796 \cdot 10^{-3} \cdot m^3/s$ |
| 141C hex | barrel per second (us;beer) | bbl/s(us;beer) | $1.173478 \cdot 10^{-1} \cdot m^3/s$ |
| 141D hex | barrel per day (us; norm fluids) | bbl/d(us;liq) | $1.380098 \cdot 10^{-6} \cdot m^3/s$ |
| 141E hex | barrel per hour (us; norm fluids) | bbl/h(us;liq) | $3.312235 \cdot 10^{-5} \cdot m^3/s$ |
| 141F hex | barrel per min (us; norm fluids) | bbl/min(us;liq) | $1.987341 \cdot 10^{-3} \cdot m^3/s$ |
| 1420 hex | barrel per second (us; norm fluids) | bbl/s(us;liq) | $1.192405 \cdot 10^{-1} \cdot m^3/s$ |
| 1421 hex | barrel per day (us; oil) | bbl/d(us;oil) | $1.840130 \cdot 10^{-6} \cdot m^3/s$ |
| 1422 hex | barrel per hour (us; oil) | bbl/h(us;oil) | $4.416313 \cdot 10^{-5} \cdot m^3/s$ |
| 1423 hex | barrel per min (us; oil) | bbl/min(us;oil) | $2.649788 \cdot 10^{-3} \cdot m^3/s$ |
| 1424 hex | barrel per second (us; oil) | bbl/s(us;oil) | $1.589873 \cdot 10^{-1} \cdot m^3/s$ |
| 1425 hex | barrel per day (us; filling tanks) | bbl/d(us;tank) | $2.409695 \cdot 10^{-6} \cdot m^3/s$ |
| 1426 hex | barrel per hour (us; filling tanks) | bbl/h(us;tank) | $5.783268 \cdot 10^{-5} \cdot m^3/s$ |
| 1427 hex | barrel per min (us; filling tanks) | bbl/min(us;tank) | $3.469961 \cdot 10^{-3} \cdot m^3/s$ |
| 1428 hex | barrel per second (us; filling tanks) | bbl/s(us;tank) | $2.081977 \cdot 10^{-1} \cdot m^3/s$ |
| 1429 hex | cubic centimeter per day | $cm^3/d$ | $1.157407 \cdot 10^{-11} \cdot m^3/s$ |
| 142A hex | cubic centimeter per hour | $cm^3/h$ | $2.777778 \cdot 10^{-10} \cdot m^3/s$ |
| 142B hex | cubic centimeter per minute | $cm^3/min$ | $1.666667 \cdot 10^{-8} \cdot m^3/s$ |
| 142C hex | cubic centimeter per second | $cm^3/s$ | $10^{-6} \cdot m^3/s$ |
| 142D hex | cubic decimeter per day | $dm^3/d$ | $1.157407 \cdot 10^{-8} \cdot m^3/s$ |
| 142E hex | cubic decimeter per hour | $dm^3/h$ | $2.777778 \cdot 10^{-7} \cdot m^3/s$ |
| 142F hex | cubic decimeter per minute | $dm^3/min$ | $1.666667 \cdot 10^{-5} \cdot m^3/s$ |
| 1430 hex | cubic decimeter per second | $dm^3/s$ | $10^{-3} \cdot m^3/s$ |
| 1431 hex | cubic meter per day | $m^3/d$ | $1.157407 \cdot 10^{-5} \cdot m^3/s$ |
| 1432 hex | cubic meter per hour | $m^3/h$ | $2.777778 \cdot 10^{-4} \cdot m^3/s$ |
| 1433 hex | cubic meter per minute | $m^3/min$ | $1.666667 \cdot 10^{-2} \cdot m^3/s$ |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1434 hex | gallon per day (us) | gal/d(us) | $4.381264 \cdot 10^{-8} \cdot m^3/s$ |
| 1435 hex | gram per day | g/d | $1.157407 \cdot 10^{-8} \cdot kg/s$ |
| 1436 hex | gram per hour | g/h | $2.777778 \cdot 10^{-7} \cdot kg/s$ |
| 1437 hex | gram per second | g/s | $10^{-3} \cdot kg/s$ |
| 1438 hex | hectoliter per day | hl/d | $1.157407 \cdot 10^{-6} \cdot m^3/s$ |
| 1439 hex | hectoliter per hour | hl/h | $2.777778 \cdot 10^{-5} \cdot m^3/s$ |
| 143A hex | hectoliter per minute | hl/min | $1.666667 \cdot 10^{-3} \cdot m^3/s$ |
| 143B hex | hectoliter per second | hl/s | $10^{-1} \cdot m^3/s$ |
| 143C hex | barrel per day (imperial; oil) | bbl/d(imp;oil) | |
| 143D hex | barrel per hour (imperial; oil) | bbl/h(imp;oil) | |
| 143E hex | barrel per minute (imperial; oil) | bbl/min(imp;oil) | |
| 143F hex | barrel per second (imperial; oil) | bbl/s(imp;oil) | |
| 1440 hex | gallon per day (imperial) | gal/d(imp) | $4.381264 \cdot 10^{-8} \cdot$ m3/s |
| 1441 hex | gallon per hour (imperial) | gal/h(imp) | $1.262803 \cdot 10^{-6} \cdot m^3/s$ |
| 1442 hex | gallon per minute (imperial) | gal/min(imp) | $7.576819 \cdot 10^{-5} \cdot m^3/s$ |
| 1443 hex | gallon per second (imperial) | gal/s(imp) | $4.546092 \cdot 10^{-3} \cdot m^3/s$ |
| 1444 hex | kilogram per day | kg/d | $1.157407 \cdot 10^{-5} \cdot kg/s$ |
| 1445 hex | kilogram per minute | kg/min | $1.666667 \cdot 10^{-2} \cdot kg/s$ |
| 1446 hex | liter per day | l/d | $1.157407 \cdot 10^{-8} \cdot$ m3/s |
| 1447 hex | megagallon per day (us; liquid) | US.liq.Mgal/d | $4.381264 \cdot 10^{-5} \cdot m^3/s$ |
| 1448 hex | megagallon per hour (us; liquid) | US.liq.Mgal/h | $1.051503 \cdot 10^{-3} \cdot m^3/s$ |
| 1449 hex | megagallon per minute (us; liquid) | US.liq.Mgal/min | $6.309196 \cdot 10^{-2} \cdot m^3/s$ |
| 144A hex | megagallon per second (us; liquid) | US.liq.Mgal/s | $3.785412 \cdot m^3/s$ |
| 144B hex | megagallon per day (imperial) | Mgal/d(imp) | $4.381264 \cdot 10^{-5} \cdot m^3/s$ |
| 144C hex | megagallon per hour (imperial) | Mgal/h(imp) | $1.262803 \cdot 10^{-3} \cdot m^3/s$ |
| 144D hex | megagallon per minute (imperial) | Mgal/min(imp) | $7.576819 \cdot 10^{-2} \cdot m^3/s$ |
| 144E hex | megagallon per second (imperial) | Mgal/s(imp) | $4.546092 \cdot m^3/s$ |
| 144F hex | milliliter per day | ml/d | $1.157407 \cdot 10^{-11} \cdot m^3/s$ |
| 1450 hex | normal cubic meter per day | $Nm^3/d$ | |
| 1451 hex | normal cubic meter per hour | $Nm^3/h$ | |
| 1452 hex | normal cubic meter per minute | $Nm^3/min$ | |
| 1453 hex | normal cubic meter per second | $Nm^3/s$ | |
| 1454 hex | normal liter per day | Nl/d | |
| 1455 hex | normal liter per hour | Nl/h | |
| 1456 hex | normal liter per minute | Nl/min | |
| 1457 hex | normal liter per second | Nl/s | |
| 1458 hex | ounce per day | oz/d | $3.281195 \cdot 10^{-7} \cdot kg/s$ |
| 1459 hex | ounce per hour | oz/h | $7.874868 \cdot 10^{-6} \cdot kg/s$ |
| 145A hex | ounce per minute | oz/min | $4.724921 \cdot 10^{-4} \cdot kg/s$ |
| 145B hex | ounce per second | oz/s | $2.834952 \cdot 10^{-2} \cdot kg/s$ |
| 145C hex | pound per day | lb/d | $5.249912 \cdot 10^{-6} \cdot kg/s$ |
| 145D hex | standard cubic meter per day | $Sm^3/d$ | |
| 145E hex | standard cubic meter per hour | $Sm^3/h$ | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 145F hex | standard cubic meter per minute | $Sm^3/min$ | |
| 1460 hex | standard cubic meter per second | $Sm^3/s$ | |
| 1461 hex | metric ton per day | t/d | $1.157407 \cdot 10^{-2} \cdot kg/s$ |
| 1462 hex | metric ton per hour | t/h | $2.777778 \cdot 10^{-1} \cdot kg/s$ |
| 1463 hex | metric ton per minute | t/min | $1.666667 \cdot 10^{1} \cdot kg/s$ |
| 1464 hex | metric ton per second | t/s | $10^{3} \cdot kg/s$ |
| 1465 hex | cubic foot per day | $ft^3/d$ | $3.277413 \cdot 10^{-7} \cdot m^3/s$ |
| 1466 hex | cubic foot per hour | $ft^3/h$ | $7.865791 \cdot 10^{-6} \cdot m^3/s$ |
| 1467 hex | cubic foot per second | $ft^3/s$ | $2.831685 \cdot 10^{-2} \cdot m^3/s$ |
| 1468 hex | kilogallon per day (us; liquid) | US.liq.kgal/d | |
| 1469 hex | kilogallon per hour (us; liquid) | US.liq.kgal/h | |
| 146A hex | kilogallon per minute (us; liquid) | US.liq.kgal/min | |
| 146B hex | kilogallon per second (us; liquid) | US.liq.kgal/s | |
| 146C hex | Standard cubic foot per second | scfs | |
| 146D hex | Standard cubic foot per minute | scfm | |
| 146E hex | Standard cubic foot per hour | scfh | |
| 146F hex | Standard cubic foot per day | scfd | |
| 1470 hex | Actual cubic foot per second | acfs | |
| 1471 hex | Actual cubic foot per minute | acfm | |
| 1472 hex | Actual cubic foot per hour | acfh | |
| 1473 hex | Actual cubic foot per day | acfd | |
| 1474 hex | long ton per day | LTon/d | $1.175980 \cdot 10^{-2} \cdot kg/s$ |
| 1475 hex | long ton per hour | LTon/h | $2.822353 \cdot 10^{-1} \cdot kg/s$ |
| 1476 hex | long ton per minute | LTon/min | $1.693412 \cdot 10^{1} \cdot kg/s$ |
| 1477 hex | long ton per second | LTon/s | $1.016047 \cdot 10^{3} \cdot kg/s$ |
| 1478 hex | barrel per day (imperial; beer) | bbl/d(imp;beer) | $1.894205 \cdot 10^{-6} \cdot m^3/s$ |
| 1479 hex | barrel per hour (imperial; beer) | bbl/h(imp;beer) | $4.546092 \cdot 10^{-5} \cdot m^3/s$ |
| 147A hex | barrel per minute (imperial; beer) | bbl/min(imp;beer) | $2.727665 \cdot 10^{-3} \cdot m^3/s$ |
| 147B hex | barrel per second (imperial; beer) | bbl/s(imp;beer) | $1.636593 \cdot 10^{-1} \cdot m^3/s$ |
| 147C hex | ounce per day (us; fluid) | fl oz/d(us) | $3.422766 \cdot 10^{-10} \cdot m^3/s$ |
| 147D hex | ounce per hour (us; fluid) | fl oz/h(us) | $8.214639 \cdot 10^{-9} \cdot m^3/s$ |
| 147E hex | ounce per minute (us; fluid) | fl oz/min(us) | $4.928784 \cdot 10^{-7} \cdot m^3/s$ |
| 147F hex | ounce per second (us; fluid) | fl oz/s(us) | $2.957270 \cdot 10^{-5} \cdot m^3/s$ |
| 1480 hex | Short ton per day | STon/d | $1.049982 \cdot 10^{-2} \cdot kg/s$ |
| 1481 hex | Short ton per hour | STon/h | $2.519958 \cdot 10^{-1} \cdot kg/s$ |
| 1482 hex | Short ton per minute | STon/min | $1.511975 \cdot 10^{1} \cdot kg/s$ |
| 1483 hex | Short ton per second | STon/s | $9.071847 \cdot 10^{2} \cdot kg/s$ |
| 1484-14FF hex | reserved for future standardization | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## D-2.8 Acceleration

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1500 hex | meter per second squared | $m/s^2$ | $m/s^2$ |
| 1501 hex | foot per second squared | $ft/s^2$ | $3.048 \cdot 10^{-1} \cdot m/s^2$ |
| 1502 hex | inch per second squared | $in/s^2$ | $2.54 \cdot 10^{-2} \cdot m/s^2$ |
| 1503 hex | angular acceleration | $rad/s^2$ | |
| 1504 hex | acceleration during free fall (gravity) | $g_n$ | $9.80665 \bullet m/s^2$ |
| 1505-15FF hex | reserved for future standardization | | |

## D-2.9 Amount of Substance

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1600 hex | mole [#11] | mol | mol |
| 1601 hex | Refer to section D-2.34 | | |
| 1602-16FF hex | reserved for future standardization | | |

## D-2.10 Angle

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1700 hex | radian (plane angle) [#12] | rad | rad |
| 1701 hex | steradian (solid angle) [#13] | sr | sr |
| 1702 hex | revolution | r | $6.283185 \bullet rad$ |
| 1703 hex | degree | ° | $(\pi/180) \bullet rad$ |
| 1704 hex | arc minute (minute) | ‘ | (1/60)° |
| 1705 hex | arc second (second) | “ | (1/60)‘ |
| 1706-17FF hex | reserved for future standardization | | |

## D-2.11 Area

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1800 hex | square meter | $m^2$ | $m^2$ |
| 1801 hex | square centimeter | $cm^2$ | $10^{-4} \cdot m^2$ |
| 1802 hex | square kilometer | $km^2$ | $10^6 \cdot m^2$ |
| 1803 hex | square yard | $yd^2$ | $0.836127 \cdot m^2$ |
| 1804 hex | square foot | $ft^2$ | $9.290304 \cdot 10^{-2} \cdot m^2$ |
| 1805 hex | square inch | $in^2$ | $6.4516 \cdot 10^{-4} \cdot m^2$ |
| 1806 hex | square mile | $mi^2$ | $2.589988 \cdot 10^6 \cdot m^2$ |
| 1807-18FF hex | reserved for future standardization | | |

[11] The mole is an SI base unit, defined as the amount of substance of a system which contains as many elementary entities as there are atoms in 0.012 kilogram of carbon 12. When the mole is used, the elementary entities must be specified and may be atoms, molecules, ions, electrons, other particles, or specified groups of such particles.
[12] The radian is defined as the plane angle between two radii of a circle that cuts off on the circumference an arc equal in length to the radius.
[13] The steradian is defined as the solid angle that, having its vertex in the center of a sphere, cuts off an area of the surface of the sphere equal to that of a square with sides of length equal to the radius of the sphere.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## D-2.12 Capacitance

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1900 hex | farad | F | $m^{-2} \bullet kg^{-1} \bullet s^{4} \bullet A^{2}$ |
| 1901 hex | millifarad | mF | $10^{-3} \bullet F$ |
| 1902 hex | microfarad | µF | $10^{-6} \bullet F$ |
| 1903 hex | nanofarad | nF | $10^{-9} \bullet F$ |
| 1904 hex | picofarad | pF | $10^{-12} \bullet F$ |
| 1905 hex | femtofarad | fF | $10^{-15} \bullet F$ |
| 1906 hex | permittivity | F/m | F/m |
| 1907-19FF hex | reserved for future standardization | | |

## D-2.13 Charge (Electric Charge)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1A00 hex | coulomb | C | $A \bullet s$ |
| 1A01 hex | millicoulomb | mC | $10^{-3} \bullet C$ |
| 1A02 hex | microcoulomb | µC | $10^{-6} \bullet C$ |
| 1A03 hex | nanocoulomb | nC | $10^{-9} \bullet C$ |
| 1A04 hex | picocoulomb | pC | $10^{-12} \bullet C$ |
| 1A05 hex | femtocoulomb | fC | $10^{-15} \bullet C$ |
| 1A06 hex | ampere hour | $A \bullet h$ | $3600 \bullet C$ |
| 1A07 hex | exposure (x and γ rays) | C/kg | $kg^{-1} \bullet C$ |
| 1A08-1AFF hex | reserved for future standardization | | |

## D-2.14 Conductance (Electric Conductance)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1B00 hex | siemens | S | $m^{-2} \bullet kg^{-1} \bullet s^{3} \bullet A^{2}$ |
| 1B01 hex | millisiemens | mS | $10^{-3} \bullet S$ |
| 1B02 hex | microsiemens | µS | $10^{-6} \bullet S$ |
| 1B03 hex | nanosiemens | nS | $10^{-9} \bullet S$ |
| 1B04 hex | picosiemens | pS | $10^{-12} \bullet S$ |
| 1B05 hex | femtosiemens | fS | $10^{-15} \bullet S$ |
| 1B06-1BFF hex | reserved for future standardization | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## D-2.15 Current (Electric Current)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1C00 hex | ampere [#14] | A | A |
| 1C01 hex | 100 milliamp, deciamp | 100mA | $10^{-1} \bullet A$ |
| 1C02 hex | milliamp | mA | $10^{-3} \bullet A$ |
| 1C03 hex | microamp | µA | $10^{-6} \bullet A$ |
| 1C04 hex | nanoamp | nA | $10^{-9} \bullet A$ |
| 1C05 hex | picoamp | pA | $10^{-12} \bullet A$ |
| 1C06 hex | femtoamp | fA | $10^{-15} \bullet A$ |
| 1C07 hex | kiloamp | kA | $10^{3} \bullet A$ |
| 1C08 hex | megaamp | MA | $10^{6} \bullet A$ |
| 1C09 hex | gigaamp | GA | $10^{9} \bullet A$ |
| 1C0A hex | magnetic field strength | A/m | |
| 1C0B-1CFF hex | reserved for future standardization | | |

## D-2.16 Energy (Heat, Work)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1D00 hex | joule | J | $m^2 \bullet kg \bullet s^{-2}$ |
| 1D01 hex | watt second | W • s | J |
| 1D02 hex | watt hour | W • h | $3.6 \bullet 10^{3} \bullet J$ |
| 1D03 hex | kilowatt hour | kW • h | $3.6 \bullet 10^{6} \bullet J$ |
| 1D11 hex | megawatt hour | MW • h | $3.6 \bullet 10^{9} \bullet J$ |
| 1D12 hex | gigawatt hour | GW • h | $3.6 \bullet 10^{12} \bullet J$ |
| 1D13 hex | terawatt hour | TW • h | $3.6 \bullet 10^{15} \bullet J$ |
| 1D04 hex | calorie (thermochemical, nutrition) | cal | $4.184 \bullet J$ |
| 1D05 hex | British thermal unit | Btu | $1055.056 \bullet 10^{3} \bullet J$ |
| 1D06 hex | kiloBtu | kBtu | $1055.056 \bullet 10^{6} \bullet J$ |
| 1D07 hex | megaBtu | MBtu | $1055.056 \bullet 10^{9} \bullet J$ |
| 1D08 hex | foot pound-force (foot-pound) | ft • lbf | $1.355818 \bullet J$ |
| 1D09 hex | electronvolt [#15] | eV | $1.60217733 \bullet 10^{-19} \bullet J$ |
| 1D0A hex | heat capacity (entropy) | J/K | J/K |
| 1D0B hex | British thermal unit (international) per degree Fahrenheit | Btu(it)/°F | $1.899101 \bullet 10^{3} \bullet J/K$ |
| 1D0C hex | specific heat capacity (specific entropy) | J/(kg • K) | J/(kg • K) |
| 1D0D hex | specific energy | J/kg | J/kg |
| 1D0E hex | energy density | $J/m^3$ | $J/m^3$ |
| 1D0F hex | molar energy | J/mol | J/mol |

[14] The ampere is an SI base unit, defined as the constant current which, if maintained in two straight parallel conductors of infinite length, of negligible circular cross section, and placed 1 meter apart in vacuum, would produce between these conductors a force equal to $2 \times 10^{-7}$ newton per meter of length.

[15] The electronvolt is a unit which is accepted for use with SI units. However, the accepted value for 1eV has been experimentally obtained and the base units shown does not represent an exact value.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1D10 hex | molar entropy (molar heat capacity) | J/(mol • K) | J/(mol • K) |
| 1D14 hex | kilowatt hour per kilogram | kW • h / kg | $3.6 \cdot 10^3 \cdot$ J/kg |
| 1D15 hex | kilojoule per kilogram | kJ/kg | $10^3 \cdot$ J/kg |
| 1D16 hex | megajoule per kilogram | MJ/kg | $10^6 \cdot$ J/kg |
| 1D17 hex | kilocalorie per kilogram | kcal/kg | $4.184 \cdot 10^3 \cdot$ J/kg |
| 1D18 hex | british thermal unit (thermochemical) per pound | Btu(th)/lb | $2.324444 \cdot 10^3 \cdot$ J/kg |
| 1D19 hex | british thermal unit (thermochemical) | Btu(th) | 1.054 • J |
| 1D1A hex | British thermal unit (thermochemical) per degree Fahrenheit | Btu(th)/°F | $1.897830 \cdot 10^3 \cdot$ J/K |
| 1D1B-1DFF hex | reserved for future standardization | | |

## D-2.17 Force

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1E00 hex | newton | N | $m \bullet kg \bullet s^{-2}$ |
| 1E01 hex | surface tension | N/m | N/m |
| 1E02 hex | kilogram-force | kgf | 9.80665 • N |
| 1E03 hex | pound-force | lbf | 4.448 • N |
| 1E04 hex | ounce-force | ozf | $2.78 \bullet 10^{-1} \bullet$ N |
| 1E05 hex | kilonewton | kN | $10^3 \cdot$ N |
| 1E06 hex | meganewton | MN | $10^6 \cdot$ N |
| 1E07 hex | millinewton | mN | $10^{-3} \cdot$ N |
| 1E08 hex | micronewton | µN | $10^{-6} \cdot$ N |
| 1E09 hex | ounce-force per second | ozf/s | $2.78 \cdot 10^{-1} \cdot$ N/s |
| 1E0A hex | ounce-force per minute | ozf/min | $4.633333 \cdot 10^{-3} \cdot$ N/s |
| 1E0B hex | ounce-force per hour | ozf/h | $7.722222 \cdot 10^{-5} \cdot$ N/s |
| 1E0C hex | ounce-force per day | ozf/d | $3.217593 \cdot 10^{-6} \cdot$ N/s |
| 1E0D-1EFF hex | reserved for future standardization | | |

## D-2.18 Frequency (includes RPM) [#16]

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1F00 hex | hertz | Hz | $s^{-1}$ |
| 1F01 hex | kilohertz | kHz | $10^3 \bullet s^{-1}$ |
| 1F02 hex | megahertz | MHz | $10^6 \bullet s^{-1}$ |
| 1F03 hex | gigahertz | GHz | $10^9 \bullet s^{-1}$ |
| 1F04 hex | counts per second | CPS | $s^{-1}$ |
| 1F05 hex | counts per millisecond | | $10^3 \bullet s^{-1}$ |
| 1F06 hex | counts per microsecond | | $10^6 \bullet s^{-1}$ |
| 1F07 hex | counts per minute | CPM | $s^{-1}$ / 60 |
| 1F08 hex | counts per hour | | $s^{-1}$ / 3600 |

[16] For the purposes of this appendix, *Frequency* is defined as rate of occurrence (for a specified event).


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1F09 hex | counts per day | | $s^{-1}$ / 86400 |
| 1F0A hex | baud rate | baud [#17] | |
| 1F0B hex | kbaud | kbaud | |
| 1F0C hex | Mbaud | Mbaud | |
| 1F0D hex | angular velocity | rad/s | |
| 1F0E hex | revolution per second (rps) | rps [#18] | |
| 1F0F hex | revolution per minute (rpm) | rpm [#19] | $1.047198 \bullet 10^{-1} \bullet$ rad/s |
| 1F10 hex | revolution per hour | r/h | |
| 1F11 hex | revolution per day | r/d | |
| 1F12-1FFF hex | reserved for future standardization | | |

[17] Usage of the symbols *Bd* and *bd* is also acceptable.
[18] Usage of the symbols *RPS* and *r/s* is also acceptable.
[19] Usage of the symbols *RPM* and *r/min* is also acceptable.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## D-2.19 Inductance

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2000 hex | henry | H | $m^2 \bullet kg \bullet s^{-2} \bullet A^{-2}$ |
| 2001 hex | millihenry | mH | $10^{-3} \bullet H$ |
| 2002 hex | microhenry | µH | $10^{-6} \bullet H$ |
| 2003 hex | nanohenry | nH | $10^{-9} \bullet H$ |
| 2004 hex | picohenry | pH | $10^{-12} \bullet H$ |
| 2005 hex | femtohenry | fH | $10^{-15} \bullet H$ |
| 2006 hex | permeability | H/m | H/m |
| 2007-20FF hex | reserved for future standardization | | |

## D-2.20 Inertia

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2100 hex | inertia (as $kg \bullet m^2$) | $kg \bullet m^2$ | $kg \bullet m^2$ |
| 2101 hex | inertia (as $mg \bullet m^2$) | $mg \bullet m^2$ | $10^{-6} \bullet kg \bullet m^2$ |
| 2102-21FF hex | reserved for future standardization | | |

## D-2.21 Length (Distance, Displacement)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2200 hex | meter [#20] | m | m |
| 2201 hex | kilometer | km | $10^3 \bullet m$ |
| 2202 hex | centimeter | cm | $10^{-2} \bullet m$ |
| 2203 hex | millimeter | mm | $10^{-3} \bullet m$ |
| 2204 hex | micron (micrometer) | µm | $10^{-6} \bullet m$ |
| 2205 hex | nanometer | nm | $10^{-9} \bullet m$ |
| 2206 hex | angstrom | Å [#21] | $10^{-10} \bullet m$ |
| 2207 hex | inch | in | $2.54 \bullet 10^{-2} \bullet m$ |
| 2208 hex | foot | ft | $3.048 \bullet 10^{-1} \bullet m$ |
| 2209 hex | yard | yd | $9.144 \bullet 10^{-1} \bullet m$ |
| 220A hex | mile | mi | $1.609344 \bullet 10^3 \bullet m$ |
| 220B hex | nautical mile | NM [#22] | $1.852 \bullet 10^3 \bullet m$ |
| 220C hex | astronomical unit | AU | $1.495979 \bullet 10^{11} \bullet m$ |
| 220D hex | light year | l.y. | $9.46073 \bullet 10^{15} \bullet m$ |
| 220E hex | parsec | pc | $3.085678 \bullet 10^{16} \bullet m$ |
| 220F hex | point (computer) | | 3.527777 • 10-4 • m [#23] |
| 2210 hex | point (printer's) | | $3.514598 \bullet 10^{-4} \bullet m$ |

[20] The meter is an SI base unit, defined as the length of the path traveled by light in vacuum during a time interval of 1/299792458 second.
[21] The capital letter "A" can be used as an alternative to the "Å" as the symbol for Angstrom. Care should be taken to avoid confusion when "A" is used as the symbol for Ampere.
[22] Care should be taken to avoid confusion with the symbol for nanometer.
[23] This is equal to (1/72) inch.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2211 hex | decimeter | dm | $10^{-1} \bullet m$ |
| 2212-22FF hex | reserved for future standardization | | |

## D-2.22 Light (Luminous Intensity)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2300 hex | candela [#24] | cd | cd |
| 2301 hex | luminance | $cd/m^2$ | $cd/m^2$ |
| 2302 hex | luminous flux (lumen) | lm | cd • sr |
| 2303 hex | illuminance (lux, lumen/$m^2$) | lx | $m^{-2} \bullet cd \bullet sr$ |
| 2304 hex | footcandle | $lm/ft^2$ | $10.76391 \bullet 10^1 \bullet lx$ |
| 2305-23FF hex | reserved for future standardization | | |

## D-2.23 Magnetic Flux

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2400 hex | weber | Wb | V • s |
| 2401 hex | magnetic flux density (tesla) | T | $V \bullet s/m^2$ |
| 2402 hex | magnetic field (oersted) | Oe | $7.957747 \bullet 10^1 \bullet A/m$ |
| 2403-24FF hex | reserved for future standardization | | |

## D-2.24 Mass

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2500 hex | kilogram [#25] | kg | kg |
| 2501 hex | gram | g | $10^{-3} \bullet kg$ |
| 2502 hex | milligram | mg | $10^{-6} \bullet kg$ |
| 2503 hex | metric ton (megagram) | t | $10^3 \bullet kg$ |
| 2504 hex | ounce (avoirdupois) | oz | $2.834952 \bullet 10^{-2} \bullet kg$ |
| 2505 hex | pound (avoirdupois) | lb | $4.535924 \bullet 10^{-1} \bullet kg$ |
| 2506 hex | short ton (2000 lb) | STon | $9.071847 \bullet 10^2 \bullet kg$ |
| 2507 hex | long ton | LTon | $1.016047 \bullet 10^3 \bullet kg$ |
| 2508-25FF hex | reserved for future standardization | | |

[24] The candela is an SI base unit, defined as the luminous intensity, in a given direction, of a source that emits monchro-matic radiation of frequency $540 \bullet 10^{12}$ hertz and that has a radiant intensity in that direction of (1/683) watt per steradian.

[25] The kilogram is an SI base unit, defined as the mass of the international prototype of the kilogram.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## D-2.25 Power (Wattage)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2600 hex | watt | W | $m^2 \bullet kg \bullet s^{-3}$ |
| 2601 hex | milliwatt | mW | $10^{-3} \bullet W$ |
| 2602 hex | microwatt | µW | $10^{-6} \bullet W$ |
| 2603 hex | nanowatt | nW | $10^{-9} \bullet W$ |
| 2604 hex | picowatt | pW | $10^{-12} \bullet W$ |
| 2605 hex | femtowatt | fW | $10^{-15} \bullet W$ |
| 2606 hex | kilowatt | kW | $10^{3} \bullet W$ |
| 2607 hex | British thermal unit (international) per hour | Btu(it)/h | $2.930711 \bullet 10^{-1} \bullet W$ |
| 260F hex | megawatt | MW | $10^{6} \bullet W$ |
| 2610 hex | gigawatt | GW | $10^{9} \bullet W$ |
| 2611 hex | terawatt | TW | $10^{12} \bullet W$ |
| 2608 hex | erg per second | erg/s | $10^{-7} \bullet W$ |
| 2609 hex | horsepower (electric, automotive) | | $7.46 \bullet 10^{2} \bullet W$ |
| 260A hex | horsepower (boiler) | | $9.8095 \bullet 10^{3} \bullet W$ |
| 260B hex | British thermal unit (international) per square foot hour | $Btu(it)/(ft^2 \bullet h)$ | $3.154591 \bullet W/m^2$ |
| 260C hex | radiant intensity | W/sr | W/sr |
| 260D hex | radiance | $W/(m^2 \bullet sr)$ | $W/(m^2 \bullet sr)$ |
| 260E hex | thermal conductivity | $W/(m \bullet K)$ | $W(m \bullet K)$ |
| 2612 hex | foot pound per minute | $ft \bullet lb/min$ | $2.259696 \bullet 10^{-2} \bullet W$ |
| 2612 hex | British thermal unit (thermochemical) per square foot hour | $Btu(th)/(ft2 \bullet h)$ | $3.152480 \bullet W/m^2$ |
| 2614 hex | British thermal unit (thermochemical) per day | Btu(th)/d | $1.220313 \bullet 10^{-2} \bullet W$ |
| 2615 hex | British thermal unit (thermochemical) per hour | Btu(th)/h | $2.92875 \bullet 10^{-1} \bullet W$ |
| 2616 hex | British thermal unit (thermochemical) per minute | Btu(th)/min | $1.75725 \bullet 10 \bullet W$ |
| 2617 hex | British thermal unit (thermochemical) per second | Btu(th)/s | $1.05435 \bullet 10^{3} \bullet W$ |
| 2618-26FF hex | reserved for future standardization | | |

## D-2.26 Radiology

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2700 hex | activity of a radionuclide (becquerel) | Bq | $s^{-1}$ |
| 2701 hex | absorbed dose (gray) | Gy | $m^2 \bullet s^{-2}$ |
| 2702 hex | dose equivalent (sievert) | Sv | $m^2 \bullet s^{-2}$ |
| 2703 hex | absorbed dose rate | Gy/s | $m^2 \bullet s^{-3}$ |
| 2704 hex | curie | Ci | $3.7 \bullet 10^{10} \bullet Bq$ |
| 2705 hex | roentgen | R | $2.58 \bullet 10^{-4} \bullet C/kg$ |
| 2706 hex | rad | rd | $10^{-2} \bullet Gy$ |
| 2707 hex | rad equivalent man | rem | $10^{-2} \bullet Sv$ |
| 2708-27FF hex | reserved for future standardization | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## D-2.27 Resistance (Electric Resistance)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2800 hex | ohm | Ω | $m^2 \bullet kg \bullet s^{-3} \bullet A^{-2}$ |
| 2801 hex | milliohm | mΩ | $10^{-3} \bullet \Omega$ |
| 2802 hex | microohm | μΩ | $10^{-6} \bullet \Omega$ |
| 2803 hex | nanoohm | nΩ | $10^{-9} \bullet \Omega$ |
| 2804 hex | picoohm | pΩ | $10^{-12} \bullet \Omega$ |
| 2805 hex | femtoohm | fΩ | $10^{-15} \bullet \Omega$ |
| 2806 hex | kiloohm | kΩ | $10^{3} \bullet \Omega$ |
| 2807 hex | megaohm | MΩ | $10^{6} \bullet \Omega$ |
| 2808 hex | gigaohm | GΩ | $10^{9} \bullet \Omega$ |
| 2809 hex | ohm centimeter | Ω • cm | $10^{-2} \bullet \Omega \bullet m$ |
| 280A hex | nanoohm meter | nΩ • m | $10^{-9} \bullet \Omega \bullet m$ |
| 280B hex | microohm meter | μΩ • m | $10^{-6} \bullet \Omega \bullet m$ |
| 280C hex | milliohm meter | mΩ • m | $10^{-3} \bullet \Omega \bullet m$ |
| 280D hex | ohm meter | Ω • m | Ω • m |
| 280E hex | kiloohm meter | kΩ • m | $10^{3} \bullet \Omega \bullet m$ |
| 280F hex | megaohm meter | MΩ • m | $10^{6} \bullet \Omega \bullet m$ |
| 2810 hex | gigaohm meter | GΩ • m | $10^{9} \bullet \Omega \bullet m$ |
| 2811 hex | kiloohm centimeter | kΩ • cm | 10 • Ω • m |
| 2812 hex | megaohm centimeter | MΩ • cm | $10^{4} \bullet \Omega \bullet m$ |
| 2813-28FF hex | reserved for future standardization | | |

## D-2.28 Sound

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2900 hex | bel | B | B |
| 2901 hex | decibel | dB | $10^{-1} \bullet B$ |
| 2902 hex | Intensity ($W \bullet m^{-2}$) | I | I |
| 2903 hex | Sound Intensity Level | IL | see [#26] |
| 2904 hex | Sound Pressure Level | SPL | see [#27] |
| 2905 hex | Voltage Level re 1mW across 600Ω | dB(mW) | see [#28] |
| 2906 hex | Voltage Level re 1 V | dB(V) | see [#29] |
| 2907 hex | Voltage Level re 1 μV | dB(μV) | |
| 2908-29FF hex | reserved for future standardization | | |

[26] *Sound Intensity Level* is $10 \bullet \log \bullet ( I/I_o )$, where $I_o = 10^{-12}$ W/m$^2$ in air and 6.66 $10^{-19}$ W/m$^2$ in water.
[27] *Sound Pressure Level* is $20 \bullet \log \bullet ( P/P_o )$, where $P_o$ = 20 μPa in air and 1 μPa in water.
[28] *Voltage Level re 1mW across 600Ω* is $20 \bullet \log \bullet ( V/.775 )$, when the reference impedance is 600Ω.
[29] *Voltage Level re 1 V* is $20 \bullet \log \bullet ( V/1 )$, where the impedance ratio is generally ignored.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## D-2.29 Torque (Moment of Force)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2A00 hex | torque (moment of force) | N • m | N • m |
| 2A01 hex | torque (mili) | $10^{-3}$ • N • m | $10^{-3}$ • N • m |
| 2A02 hex | dyne centimeter | dyn • cm | $10^{-7}$ • N • m |
| 2A03 hex | kilogram-force meter | kgf • m | 9.80665 • N • m |
| 2A04 hex | ounce-force inch (ounce-inch) | ozf • in | 7.0615 • $10^{-3}$ • N • m |
| 2A05 hex | pound-force inch (pound-inch) | lbf • in | 1.1298 • $10^{-1}$ • N • m |
| 2A06 hex | pound-force foot (pound-foot) | lbf • ft | |
| 2A07 hex | torque per ampere | N • m/A | N • m/A |
| 2A08 hex | torque per ampere (mili) | mN • m/A | $10^{-3}$ • N • m/A |
| 2A09-2AFF hex | reserved for future standardization | | |

## D-2.30 Velocity (Speed) [#30]

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2B00 hex | meter per second | m/s | m/s |
| 2B01 hex | centimeter per second | cm/s | $10^{-2}$ • m/s |
| 2B02 hex | kilometer per hour | km/h | 2.7777 • $10^{-1}$ • m/s |
| 2B03 hex | speed of light | c | 3 • $10^{8}$ • m/s |
| 2B04 hex | mile per hour | mi/h | 4.4704 • $10^{-1}$ • m/s |
| 2B05 hex | knot (nautical mile per hour) | kt | (1852/3600) • m/s |
| 2B06 hex | foot per second | ft/s | 3.0480 • $10^{-1}$ • m/s |
| 2B07 hex | inch per second | in/s | 2.5400 • $10^{-2}$ • m/s |
| 2B08 hex | millimeter per second | mm/s | $10^{-3}$ • m/s |
| 2B09 hex | millimeter per minute | mm/min | 1.6666 • $10^{-5}$ • m/s |
| 2B0A hex | millimeter per hour | mm/h | 2.7777 • $10^{-7}$ • m/s |
| 2B0B hex | millimeter per day | mm/d | 1.1574 • $10^{-8}$ • m/s |
| 2B0C hex | inch per minute | in/min | 4.2333 • $10^{-4}$ • m/s |
| 2B0D hex | inch per hour | in/h | 7.0555 • $10^{-6}$ • m/s |
| 2B0E hex | inch per day | in/d | 2.9398 • $10^{-7}$ • m/s |
| 2B0F-2BFF hex | reserved for future standardization | | |

## D-2.31 Dynamic Viscosity

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2C00 hex | dynamic viscosity | Pa • s | $m^{-1}$ • kg • $s^{-1}$ |
| 2C01 hex | poise | P | $10^{-1}$ • Pa • s |
| 2C02 hex | centipoise | cP | $10^{-3}$ • Pa • s |
| 2C03 hex | millipascal second | mPa • s | $10^{-3}$ • Pa • s |
| 2C04-2CFF hex | reserved for future standardization | | |

[30] For the purposes of this appendix, *Velocity* uses the traditional definition of distance over time, and thus units for *baud rate* and *rpm* are listed in the *Frequency* group.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## D-2.32 Voltage

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2D00 hex | volt | V | $m^2 \bullet kg \bullet s^{-3} \bullet A^{-1}$ |
| 2D01 hex | millivolt | mV | $10^{-3} \bullet V$ |
| 2D02 hex | microvolt | µV | $10^{-6} \bullet V$ |
| 2D03 hex | nanovolt | nV | $10^{-9} \bullet V$ |
| 2D04 hex | picovolt | pV | $10^{-12} \bullet V$ |
| 2D05 hex | femtovolt | fV | $10^{-15} \bullet V$ |
| 2D06 hex | kilovolt | kV | $10^{3} \bullet V$ |
| 2D07 hex | megavolt | MV | $10^{6} \bullet V$ |
| 2D08 hex | gigavolt | GV | $10^{9} \bullet V$ |
| 2D09 hex | electric field strength | V/m | V/m |
| 2D0A-2DFF hex | reserved for future standardization | | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## D-2.33 Volume

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2E01 hex | cubic meter | $m^3$ | $m^3$ |
| 2E02 hex | liter | l [#31] | $10^{-3} \bullet m^3$ |
| 2E03 hex | milliliter | ml | $10^{-6} \bullet m^3$ |
| 2E04 hex | kiloliter | kl | $m^3$ |
| 2E05 hex | cubic yard | $yd^3$ | $7.645549 \bullet 10^{-1} \bullet m^3$ |
| 2E06 hex | cubic foot | $ft^3$ | $2.831685 \bullet 10^{-2} \bullet m^3$ |
| 2E07 hex | cubic inch | $in^3$ | $1.638706 \bullet 10^{-5} \bullet m^3$ |
| 2E08 hex | gallon (U.S.) | gal(us liq) | $3.785412 \bullet 10^{-3} \bullet m^3$ |
| 2E09 hex | quart (U.S. liquid) | liq qt | $9.463529 \bullet 10^{-4} \bullet m^3$ |
| 2E0A hex | pint (U.S. liquid) | pt(us) | $4.731765 \bullet 10^{-4} \bullet m^3$ |
| 2E0B hex | ounce (U.S. fluid) | fl oz(us) | $2.957353 \bullet 10^{-5} \bullet m^3$ |
| 2E0C hex | barrel (U.S.) | bbl | $1.589873 \bullet 10^{-1} \bullet m^3$ |
| 2E0D hex | specific volume | $m^3/kg$ | $m^3/kg$ |
| 2E0E hex | barrel (us; beer) | bbl(us;beer) | $1.1924 \bullet 10^{-1} \bullet m^3$ |
| 2E0F hex | barrel (us; norm fluids) | bbl(us;liq) | $1.1924 \bullet 10^{-1} \bullet m^3$ |
| 2E10 hex | barrel (us; filling tanks) | bbl(us;tank) | $2.08198 \bullet 10^{-1} \bullet m^3$ |
| 2E11 hex | cubic centimeter | $cm^3$ | $10^{-6} \bullet m^3$ |
| 2E12 hex | cubic decimeter | $dm^3$ | $10^{-3} \bullet m^3$ |
| 2E13 hex | hectoliter | hl | $10^{-1} \bullet m^3$ |
| 2E14 hex | barrel (imperial; oil) | bbl(imp;oil) | $1.58987 \bullet 10^{-1} \bullet m^3$ |
| 2E15 hex | gallon (imperial; liquid) | gal(imp) | $4.546092 \bullet 10^{-3} \bullet m^3$ |
| 2E16 hex | megagallon (us; liquid) | US.liq.Mgal | $3.785412 \bullet 10^{3} \bullet m^3$ |
| 2E17 hex | megagallon (imperial) | Mgal(imp) | $4.546092 \bullet 10^{3} \bullet m^3$ |
| 2E18 hex | megaliter | Ml | $10^{3} \bullet m^3$ |
| 2E19 hex | normal liter | Nl | |
| 2E1A hex | standard cubic meter | $Sm^3$ | |
| 2E1B hex | kilogallon (us; liquid) | kgal(us) | $3.785412 \bullet 10^{-3} \bullet m^3$ |
| 2E1C hex | standard cubicdecimeter | $Sdm^3$ | |
| 2E1D hex | standard cubic inch | SCI | |
| 2E1E hex | standard cubic foot | SCF | |
| 2E1F hex | barrel (imperial; beer) | bbl(imp;beer) | $1.63659 \bullet 10^{-4} \bullet m^3$ |
| 2E20 hex | quart (imperial liquid) | Imp.qt. | $1.1365 \bullet 10^{-3} \bullet m^3$ |
| 2E21 hex | acre foot | af | $1.23348 \bullet 10^{3} \bullet m^3$ |
| 2E22 hex | normal cubic meter | $Nm^3$ | |
| 2E23 hex | normal cubic decimeter | $Ndm^3$ | |
| 2E24-2EFF hex | reserved for future standardization | | |

[31] According to Interpretation of the SI for the United States and Metric Conversion Policy for Federal Agencies, the letter l and the letter L are internationally accepted symbols for the unit “liter”.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

## D-2.34 Density

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 1601 hex | mole per cubic meter | $mol/m^3$ | $mol/m^3$ |
| 2F01 hex | watt per square meter | $W/m^2$ | $kg \bullet s^{-3}$ |
| 2F02 hex | joule per cubic meter | deprecated, see value 1D0E hex | |
| 2F03 hex | electric charge density | $C/m^3$ | $m^{-3} \bullet s \bullet A$ |
| 2F04 hex | electric flux density | $C/m^2$ | $m^{-2} \bullet s \bullet A$ |
| 2F05 hex | current density | $A/m^2$ | |
| 2F06 hex | British thermal unit (international) per cubic foot | $Btu(it)/ft^3$ | $3.725895 \bullet 10^4 \bullet J/m^3$ |
| 2F07 hex | mass density | $kg/m^3$ | $kg/m^3$ |
| 2F08 hex | mass density | $g/cm^3$ | $10^3 \bullet kg/m^3$ |
| 2F09 hex | counts per milliliter | 1/ml | $10^6$/m3 |
| 2F0A hex | counts per gallon (us; liquid) | 1/gal(us) | |
| 2F0B hex | pound per gallon (us; liquid) | lb/gal(us) | |
| 2F0C hex | pound per cubic foot | $lb/ft^3$ | |
| 2F0D hex | pound per cubic inch | $lb/in^3$ | |
| 2F0E hex | gram per milliliter | g/ml | $10^6 \bullet g/m^3$ |
| 2F0F hex | gram per liter | g/l | $10^3 \bullet g/m^3$ |
| 2F10 hex | kilogram per liter | kg/l | $10^6 \bullet g/m^3$ |
| 2F11 hex | microgram per liter | µg/l | $10^{-3} \bullet g/m^3$ |
| 2F12 hex | microgram per cubic meter | $\mu g/m^3$ | $10^{-6} \bullet g/m^3$ |
| 2F13 hex | gram per cubic meter | $g/m^3$ | $10^{-3}$ $kg/m^3$ |
| 2F14 hex | gram per standard cubic centimeter | $g/Scm^3$ | $10^{-9} \bullet kg/m^3$ |
| 2F15 hex | kilogram per cubic decimeter | $kg/dm^3$ | $10^3 \bullet kg/m^3$ |
| 2F16 hex | kilogram per normal cubic meter | $kg/Nm^3$ | |
| 2F17 hex | kilogram per normal liter | kg/Nl | |
| 2F18 hex | kilogram per standard cubic meter | $kg/Sm^3$ | |
| 2F19 hex | pound per barrel (us; beer) | lb/bbl(us;beer) | $3.865369 \bullet kg/m^3$ |
| 2F1A hex | pound per barrel (us; norm fluids) | lb/bbl(us;liq) | $3.804014 \bullet kg/m^3$ |
| 2F1B hex | pound per barrel (us; oil) | lb/bbl(us;oil) | $2.853010 \bullet kg/m^3$ |
| 2F1C hex | pound per barrel (us; filling tanks) | lb/bbl(us;tank) | $2.178662 \bullet kg/m^3$ |
| 2F1D hex | pound per barrel (imperial; beer) | lb/bbl(imp;beer) | $2.771565 \bullet kg/m^3$ |
| 2F1E hex | pound per barrel (imperial; oil) | lb/bbl(imp;oil) | |
| 2F1F hex | pound per gallon (imperial) | lb/gal (imp) | $9.977633 \bullet 10 \bullet kg/m^3$ |
| 2F20 hex | pound per standard cubic foot | $lb/Sft^3$ | |
| 2F21 hex | standard density at 4°C | SD4°C | $10^3 \bullet kg/m^3$ |
| 2F22 hex | standard density at 15°C | SD15°C | $9.9910 \bullet 10^2 \bullet kg/m^3$ |
| 2F23 hex | standard density at 20°C | SD20°C | $9.9820 \bullet 10^2 \bullet kg/m^3$ |
| 2F24 hex | specific gravity at 4°C | SG4°C | |
| 2F25 hex | specific gravity at 15°C | SG15°C | |
| 2F26 hex | specific gravity at 20°C | SG20°C | |
| 2F27 hex | milligram per liter | mg/l | $10^{-3} \bullet kg/m^3$ |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 2F28 hex | kilogram per liter | kg/l | $10^3 \bullet$ kg/m3 |
| 2F29 hex | part per billion (density) | ppb-dens | |
| 2F2A hex | part per million (density) | ppm-dens | |
| 2F2B hex | mole per liter | mol/l | |
| 2F2C hex | percent solid per volume | % sol/vol | |
| 2F2D hex | percent solid per weight | % sol/mass | |
| 2F2E hex | percent steam quality | % stm qual | |
| 2F2F hex | British thermal unit (thermochemical) per cubic foot | Btu(th)/ft3 | $3.72340 \bullet 10^4 \bullet J/m^3$ |
| 2F30-2FFF hex | reserved for future standardization | | |

## D-2.35 Relative Density

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 3000 hex | degree API | °API | |
| 3001-30FF hex | reserved for future standardization | | |

## D-2.36 Electric Conductivity

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 3100 hex | microsiemens per meter | µS/m | $10^{-6} \bullet$ S/m |
| 3101 hex | millisiemens per meter | mS/m | $10^{-3} \bullet$ S/m |
| 3102 hex | siemens per meter | S/m | S/m |
| 3103 hex | kilosiemens per meter | kS/m | $10^3 \bullet$ S/m |
| 3104 hex | megasiemens per meter | MS/m | $10^6 \bullet$ S/m |
| 3105 hex | nanosiemens per centimeter | nS/cm | $10^{-7} \bullet$ S/m |
| 3105 hex | microsiemens per centimeter | µS/cm | $10^{-4} \bullet$ S/m |
| 3107 hex | millisiemens per centimeter | mS/cm | $10^{-1} \bullet$ S/m |
| 3108 hex | siemens per centimeter | S/cm | $10^2 \bullet$ S/m |
| 3109 hex | microsiemens per millimeter | µS/mm | $10^{-3} \bullet$ S/m |
| 310A-31FF hex | reserved for future standardization | | |

## D-2.37 Analytics

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 3200 hex | liter per cubic meter | l/m³ | $l0^{-3} \bullet m^3/m^3$ |
| 3201 hex | mircroliter per cubic meter | µl/m³ | $10^{-9} \bullet m^3/m^3$ |
| 3202 hex | microliter per liter | µl/l | $10^{-6} \bullet m^3/m^3$ |
| 3203 hex | milliliter per cubic meter | ml/m³ | $10^{-6} \bullet m^3/m^3$ |
| 3204 hex | milliliter per liter | ml/l | $l0^{-3} \bullet m^3/m^3$ |
| 3205 hex | million cells per milliliter | MM cells/ml | |
| 3206 hex | percent saturated | %Sat | |
| 3207 hex | percent volume | %vol | |
| 3208 hex | volume percent | Vol% | |


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 3209 hex | weight percent | WT-% | |
| 320A hex | degree Balling | °Balling | |
| 320B hex | degree Baumé (heavy) | °Baum (hv) | |
| 320C hex | degree Baumé (light) | °Baum (lt) | |
| 320D hex | degree Brix | °Brix | |
| 320E hex | degree Plato | °Plato | |
| 320F hex | relative hydrogen | rH | |
| 3210 hex | gram per kilogram | g/kg | $10^{-3}$ • kg/kg |
| 3211 hex | part per million by volume | ppmvol | |
| 3212 hex | percent per kelvin | %/K | |
| 3213 hex | percent per microvolt | %/µV | |
| 3214 hex | percent per millivolt | %/mV | |
| 3215 hex | percent per volt | %/V | |
| 3216 hex | nanoampere per milligram per liter | nA/(mg/l) | |
| 3217 hex | ampere per kilogram per cubic meter | A/(kg/m^3) | |
| 3218 hex | ampere per pascal | A/Pa | |
| 3219 hex | picoampere per hectopascal | pA/hPa | |
| 321A hex | ampere per hectopascal | A/hPa | |
| 321B hex | turbidity (formazine nephelometric unit) | FNU | |
| 321C hex | turbidity (nephelometric turbidity unit) | NTU | |
| 321D hex | pascal per ampere | Pa/A | |
| 321E hex | practical salinity units | PSU | |
| 321F hex | millivolt per percent | mV/% | |
| 3220 hex | Volt per percent | V/% | |
| 3221 hex | millivolt per picohenry | mV/pH | $10^{9}$ • V/H |
| 3222 hex | volt per picohenry | V/pH | $10^{12}$ • V/H |
| 3223 hex | percent of total solids | %TS | |
| 3224 hex | picoampere per milligram per liter | pA/(mg/l) | |
| 3225 hex | kilogram per mol | kg/mol | |
| 3226 hex | gram per mol | g/mol | |
| 3227-32FF hex | reserved for future standardization | | |

## D-2.38 Kinematic Viscosity[#32]

| Value | Name | Symbol | Representation in SI Units |
|---|---|---|---|
| 3300 hex | stokes | St | $10^{-4}$ • $m^2/s$ |
| 3301 hex | centistokes | cSt | $10^{-6}$ • $m^2/s$ |
| 3302 hex | square millimeter per second | $mm^2/s$ | $10^{-6}$ • $m^2/s$ |
| 3303 hex | square meter per second | $m^2/s$ | $m^2/s$ |
| 3304-33FF hex | reserved for future standardization | | |

[32] **Kinematic viscosity** is defined as the ratio of absolute or dynamic viscosity to density. Kinematic viscosity can be obtained by dividing the absolute viscosity of a fluid with its mass density.


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)

This page is intentionally left blank


SITE SUBSCRIPTION ONLY; SUBSCRIPTION TERMS AND CONDITIONS OUTLINED ON PAGE (iii)