CIP Conformance Test Specification

PUB00166

October 2016

Acknowledgements

The original authors, Edward Polce and Patrick Maloney, wish to thank those who contributed to the development of this document. These contributors include employees of Allen-Bradley, Inc., Daniel Noonen, Jeffery Hanneman, Ronald Holl, Gregory Majcher, Douglas Stanton and many others. From the University of Warwick, U.K., Mr. Soo Beng Khoh. Also, members of the ODVA Conformance SIG who reviewed the document William Cummings, Cyrilla Dalstra, Bahman Radjabi, Takuya Shimomura, Robert Svacina and John Wozniak and other contributors.

References

CIP Networks Library, Volume 1, Common Industrial Protocol, Edition 3.21, November 2016, © 2001 through 2016 by ODVA.

CIP Networks Library, Volume 2, EtherNet/IP Adaptation of CIP, Edition 1.22, November 2016, © 1994 through 2016 by ODVA.

CIP Networks Library, Volume 3, DeviceNet Adaptation of CIP, Edition 1.14, November 2013, © 1994 through 2016 by ODVA.

Revision Control

Version	Date	Author	Notes/Description of Changes
2.0-1.0	10/15/97	E. Polce	Version 1.3-1.10 with V2.0 additions as noted. 4.1 Dup Mac Test – Added OffLine Connection Set Test 4.2 Transport Layer Test – corrected steps 35, 37, 38. 5.1 Identity Object – added attributes and Device Heartbeat, step 6.4 5.3 DeviceNet Object – corrected text for step 6.13.Child Test step 6.15.5, Deferred Delete, step 6.15.6. 5.4 Assembly Object – revised Object Services. 5.5 Connection Object – Production Inhibit Timer is conditional attribute, Added Deferred Delete step 1.7
A10	03/20/98	E. Polce	4.2 Transport Layer Test – revised step 28, added test for fragmented message = consumed connection size. 5.5 Connection Object – added Check Production Inhibit Timer/EPR error at Apply, step 2.27. 5.6 Register Object – revised Behavior Test
A11	09/20/98	E. Polce	4.1 Dup Mac Test – Added checks for req/rsp to all MacIds in Dup Mac Req Msg. to step 5 4..7 UCMM Test – Modified step 2 verification of how the message body formats are implemented. 5.3 DeviceNet Object – Revised step 5.4 to use all service codes for all devices. Revised step 6.1 to set MacId to new value, then reset. Revised step 6.7 to verify I/O clients use EPR for transmission trigger timer during pre-consumption . Step 6.13 done for all choices. Clarified step 6.15.6. Revised step 6.19.1 to allow produced size = 0. 5.4 Assembly Object – Modified Step 5.1. Added step 6.0, profile verification. 5.5 Connection Object – Revised check for default EM EPR to allow for timer tick rounding. Revised step 2.26.2 WTA Auto-Delete for Producing only connection Added 5.27 Motor Data, 5.28 Control Supervisor, 5.29 AC/DC Drive, 5.31 Overload and 5.32 Softstart Objects
A-12	03/16/99	E. Polce	4.1 Dup Mac Test – Added check for OCS Identify response to Error Test step 13.1 5.0 Persistent Verifications for Protocol Conformance Group2 Only error response added 5.3 DeviceNet Object – Revised step 6.16 for COS path 5.28 Control Supervisor Object – Revised Check for Range attribute value in Behavior Test
A-13	03/21/00	E. Polce	4.5 Group 2 Server – Added send I/O message with zero data to step2 5.2 Message Router Object – Added Set_Attribute_Single to instance 5.3 DeviceNet Object – Revised steps 6.15.4 and 6.15.5 5.13 Parameter Object – Revised step 6.2.1, Added 6.6 5.14 Parameter Group Object – Added Set_Attribute_Single, instance Added tests for Mass Flow Controller Profile and Objects. Added class attributes 1 – 7 to all objects.
A-14	02/16/01	E. Polce	4.5 Group 2 Server – Added send Idle I/O message (no data) to step 2 5.35 S-Analog Actuator Object – Revised step 6.4 for Safe Value Added tests for Vacuum/Pressure Gage Profile and Objects.

Version	Date	Author	Notes/Description of Changes
A-15	09/14/01	E. Polce	4.3 Revised Profile Verification Test 5.0 Persistent Verifications for Set Attribute Single response added 5.1 Identity – added step 6.5, multi instance test 5.9 Discrete Output Point – added Behavior test for explicit control 5.5 Connection – added produced/consumed path required if produced/consumed size is non-zero to step 2.16.1 5.12 Analog Output Point – revised Behavior test section, added test for explicit control. 5.23 Added Position Controller Supervisor tests 5.24 Added Position Controller tests. 5.38 Trip Point – revised Behavior test section, 6.1.1 and 6.1.2 to disable the trip point not tested.
A-16	03/22/02	E. Polce	5.34 S-Device Supervisor Object – Added Process Control Valve 5.35 S-Analog Sensor Object – Added Subclass 6 attribute 5.37 S-Single Stage Controller Object – Added Subclass 1 attributes and 6.8) PID & Source Select Test Added Process Control Valve Profile and 5.33 Selection Object.
A-17	07/17/03	E. Polce	5. 7 Register Object – Added step 6.4 for RF/DC Power Generator 5.28 Control Supervisor Object – Added errata attributes, step 6.4, 6.5 5. 29 AC/DC Drive Object – Added errata 5 attributes 5.34 S-Device Supervisor Object – Added Power Generator Subclass 5.37 S-Single Stage Controller Object – Added Subclasses 2, 3, 4 Added 5.40 File Object.
A-18	07/21/06	T. Weaver	Organization and formatting edits made. No content revision. DeviceNet specific material was separated from the original DevicNet Protocol Test Specification dociument. This is now the CIP Test Specification.
A-18	05/02/06	E. Polce	5. 1 Identity Object – Added Semaphore and Interface Type Attributes 5.4 Assembly – added Size attribute 5.5 Connection – added Connection_TimeOut_Multiplier attribute and Safety attributes and services 5.9 Discrete Output Point – added Test Output Mode for Safety, instance attribute 13 5.15, 5.16, 5.19, 5.20 Discrete and Analog Group Objects, added checks for attribute access and values in the bound instances. 5.35 S-Analog Sensor – added Status Extension, class attribute 32, and Value Descriptor, instance attribute 37 Added 5.40 File Object, Instance Types, EDS Upload Added Safety Objects – 5.41 Safety Supervisor, 5.42 Safety Validator, 5.43 Safety Discrete Output Point, 5.44 Safety Discrete Output Group, 5.45 Safety Discrete Input Point, 5.46 Safety Discrete Input Group, 5.47 Safety Dual Channel Output. Added Connection Confiuration Object 5.48, Port Object 5.49
A-20/A-6	01/01/09	ODVA	Connection Manager Class 3 Tests
A-21/A-7	08/04/09	ODVA	Connection Manager, TCP Close I/O tests Add Time Sync Object and Position Sensor objects Added Message Router Error tests.
A-22/A-8	12/2/10	ODVA	Connection Manager – Updated Forward Open, Unconnected_Send, Compatible Electronic Key tests; Trip Point – Added support for class revision 2; Miscellaneous formatting edits.
SA-6	9/2/11	ODVA	Safety Analog Input Point, Safety Dual-Channel Analog Input

Version	Date	Author	Notes/Description of Changes
	9/9/11	ODVA	Move CIP Safety content to PUB00170
EtherNet/IP A-9	10/12/11	ODVA	Connection Manager – ForwardOpen Status Code Revisions (PC2010-2), Large ForwardOpen tests. Add Multiple Service Packet test (from A8 Interop 1.4 spec) to Message Router object test.
SA-6	11/2/11	ODVA	Second pass at moving CIP Safety content to PUB00170
	11/21/11	ODVA	Merged from mainline by applying SA6 changes to A9 revision of document
DeviceNet CT-24	05/29/12	ODVA	Add support to scattered Identity Object Instances.
DeviceNet CT-24	05/31/12	ODVA	Add test for energy objects.
EtherNet/IP CT-10	08/08/12	ODVA	Add new behavior test for energy objects.
EtherNet/IP CT-11	04/06/13	ODVA	File Object Test Change.
EtherNet/IP CT-11	08/01/13	ODVA	Electronic Key test change.
EtherNet/IP CT-11	08/01/13	ODVA	Connection Manager connection size test update.
EtherNet/IP CT-11	09/06/13	ODVA	Added Duplicate I/O packet test in Connection Manager Object Test.
EttherNet/IP CT-11	09/13/13	ODVA	Support Time Sync Object rev 3. Added NV attribute test for this object.
EtherNet/IP CT-12 DeviceNet CT-26	11/06/14	ODVA	File Object Test update according to Vol 1 Ed. 3.16 specification update.
EtherNet/IP CT-12	11/19/14	ODVA	Added Connection Manager tests: Out-of-order I/O packets test; Forward Open Connection ID Reuse; Forward Close Wrong IP Address verification
EtherNet/IP CT-12	12/09/14	ODVA	Added documentation for Motion Device Axis Object
EtherNet/IP CT-13	03/10/15	ODVA	Added documentation for TCL Object Test
EtherNet/IP CT-13	10/26/15	ODVA	Added new I/O tests in Connection Manager Object Test
EtherNet/IP CT-13	11/03/15	ODVA	Added documentation for OCL Object Test
EtherNet/IP CT-13 DeviceNet CT-27	12/04/15	ODVA	Support Port Object rev 2
EtherNet/IP CT-13 DeviceNet CT-27	12/04/15	ODVA	Support rev 2 for three energy objects
EtherNet/IP CT-14 DeviceNet CT-28	09/23/16	ODVA	New attributes are added to support Big 12 Diagnostics.
EtherNet/IP CT-14	10/17/16	ODVA	New I/O Tests
EtherNet/IP CT-14 DeviceNet CT-28	10/24/16	ODVA	Support Power Cutailment Object and Power Management Object.

Table of Contents

1. Introduction

1.0 Purpose

This CIP Conformance Test Specification was initially prepared using the CIP Specification, Volume 1, Edition 2.2 and DeviceNet Specification Volume 3, Edition 1.1. Later revisions to the test specification reflect the CIP Specifications in effect at the time of the revision. The most recent revisions are cited in the References section (see page 2).

This document defines the following:

- CIP protocol conformance criteria.
- Individual tests used to demonstrate conformance with the CIP Specification.
- Testing process used to demonstrate conformance with the CIP Specification.

This test specification should be referenced when developing products or defining a Quality Test Plan (QTP) for products intended to comply with the CIP Specifications.

This document contains descriptions of CIP conformance tests and is intended to provide the overall guidelines to verify that a device conforms to the CIP Specifications.

This document is not a step by step guide to conformance testing for specific devices. Not all tests are applied to all products. The specific tests used to demonstrate that a device is compliant with the CIP Specification depend on the device type and the CIP objects implemented. This document provides the guidelines for determining which tests are applicable for a specific device.

The execution of conformance tests for a CIP device is expected to be done using an automated, computer based, test system. This concept is basic throughout this document and there are references to an automated process. This document provides a general description of the CIP Conformance tests. This document is not intended as a test system design document and any software implementation details of an automated test system are left to the implementers.

1.1 Audience

This document is intended for use by product engineers responsible for design, development and testing of industrial automation devices using the CIP communications protocol as defined in the CIP Specification. A good working knowledge of industrial automation technology is assumed. The reader is expected to be thoroughly familiar with the concepts described in the CIP Specifications.

1.2 Document Scope

Tests described in this document provide a means of qualifying a product as being conformant. They are neither designed nor intended to verify that product functional requirements have been achieved. Tests described in this document may qualify a significant level of CIP functionality. Conformance tests are intended to be used in addition to, not in place of, tests that verify product functional, or system, performance. When the CIP Protocol conformance test can establish the state or behavior of the device via the CIP network, that state or behavior is tested. If a state or behavior can not be established or verified via the CIP network, that state or behavior must be verified by product developer functional testing.

1.3 Features Not Included In Test Specifications

The following list of CIP device profiles, objects, and features are not covered by this test specification. If you need support for any device listed below, contact ODVA.

- Tests for the Residual Gas Analyzer, RF Power Generator, DC Power Generator, and Turbo Molecular Vacuum Pump, Fluid Flow Controller and CIP Motion Drive device profiles.
- Quick Connect behavior as defined for the DeviceNet object.
- Discrete Group, Analog Group, Group, Block Sequencer, Command Block, S-Sensor Calibration, Event Log, and Drive Axis objects.
- Selection object test is limited to the behavior defined for the Process Control Valve.
- Connection object Connection Bind and Connection Owner services
- Message Router Symbolic Translation service.

1.4 CIP Definitions

The following definitions from the DeviceNet Specification are included here for the convenience of the reader. Refer to the DeviceNet Specification for a detailed discussion of these terms.

Client	A device that is the initiator of a command or data packet transmission.
DeviceNet Master	The device that gathers and distributes I/O data for the process controller. A DeviceNet Master scans its Slave devices based on a scan list it contains. With respect to the network, the Master is a **Group 2 Client** or a **Group 2 Only Client**.
DeviceNet Slave	The device from which the DeviceNet Master gathers I/O data and to which the DeviceNet Master distributes I/O data. With the exception of the Duplicate MAC ID Check message, the slave does not initiate any communications until told to do so. With respect to the network, the Slave is a **Group 2 Server** or a **Group 2 Only Server**.
Device Profile	Part of the CIP Specification that defines a CIP Object Model, I/O data format and Configuration Interface for the device. This information is required for conformance testing.
Explicit Message Forwarding	A service performed by **Group 2 Only Clients** for their **Group 2 Only Servers**. The **Group 2 Only Client** screens for all Explicit Messages bound for a Server it owns and re-routes the message across the single Explicit Messaging connection to the Server.
Group 2 Client	A device that has gained ownership of the **Predefined Master/Slave Connection Set** in a Server such that it can act as the Client on those Predefined connections.
Group 2 Only Client	A device that acts as a Group 2 Client to a **Group 2 Only Server**. The Group 2 Only Client performs the UCMM function for its UCMM Incapable servers. It also screens for Explicit Messages bound for any of its **Group 2 Only Servers** and redirects those messages across the single Explicit Messaging connection to the Server.

Group 2 Only Client UCMM Function	This is a function performed by **Group 2 Only Clients** for their **Group 2 Only Servers**. The Client screens for all unconnected requests bound for a server it owns and returns an appropriate response for the Server.
Group 2 Only Server	A Slave (server) device that is UCMM incapable and must use the Predefined Master/Slave Connection Set. A Group 2 Only Server can receive <u>only</u> Group 2 messages, and behaves only as a Group 2 Server.
Group 2 Server	A UCMM capable device that has been told to act as the Server for the Predefined Master/Slave Connection Set connections.
Predefined Master/Slave Connection Set	A subset of the DeviceNet application layer protocol that allows product implementation using CAN based micro-controllers with limited hardware screening capabilities. The Predefined Master/Slave Connection Set is to be implemented by products intended for use in a Master/Slave architecture.
Server	A device that acts as the server for Predefined Master/Slave Connections.
Unconnected Message Manager (UCMM)	The port through which all unconnected requests enter a device. This port is represented by an identifier contained in Message Group 3.
UCMM Capable Device	A device that supports the Unconnected Message Manager (UCMM). At minimum, this requires support of the Unconnected Request Message port within Message Group 3.
UCMM <u>In</u>capable Device	Typically a low–level device that, due to network interrupt management and CAN chip screening capabilities, <u>does not</u> support the UCMM.

1.5 <u>CIP Conformance Test Definitions</u>

The following definitions are specific to the CIP Conformance Test Specification. These terms are used throughout this document.

AIG	Analog Input Group object.
AIP	Analog Input Point object.
AOG	Analog Output Group object.
AOP	Analog Output Point object.
Available	The device is powered up and ready to communicate over the CIP network. CIP objects implemented in the device may be accessed.
DIG	Discrete Input Group object.
DIP	Discrete Input Point object.
DOG	Discrete Output Group object.
DOP	Discrete Output Point object.
Error_Response	The device responds negatively to the requested CIP service with status code other than zero or (DeviceNet) Service Code x14 response, x94, a General Error Code, and an Additional Error Code, i.e. 94 08 FF, the Service_Not_Supported error.
Idle Command	The Idle command is a Bit–Strobed or Poll I/O message that contains no application data, (zero length).
No_Response	The device does not respond to the requested CIP service.

Profile Conformance	Device exhibits the characteristics and behavior required by the CIP Profile definition it implements.
Protocol Conformance	Testing of products for the characteristics and behavior required by the CIP Specifications.
Protocol Data Unit (PDU)	The minimum unit of data passed between devices. The PDU contains data (I/O, service information, etc.) and may contain protocol overhead.
Run Command	The Run command is a Bit–Strobed or Poll I/O message that contains application data.
Success_Response	A successful response to the requested CIP service. For example, x85 is the Success_Response for a Reset request (x05). x89 is the Success_Response for a Delete request (x09). This term includes data returned with the response message. For example, the Success_Response for a Get_Attribute_Single request (x0E) is x8E plus the expected data.

2. CIP Conformance Test Process

2.0 Protocol Test Overview

The number and types of tests executed depends on the type of device tested and the CIP objects implemented in the device. The test procedure requires that information specific to the Device Under Test (DUT) and CIP objects implemented in the DUT be included in the test configuration data. The Device Profile contains the information specific to the device (refer to the CIP Specification, Volume 1, for a detailed explanation of Device Profiles). Information from the Device Profile and Statement of Compliance is used to select and configure tests for the DUT. The conformance tests are applied to the DUT and a report of results is generated.

2.1 Product Definition

Definition of the DUT is the first step in the CIP Conformance Test process. Product definition requires the following information.

- Application Network Terms - Network terms indicate what functions a device performs. Devices are described using one or more of these terms.
- Device Profile - CIP devices should have a Device Profile that defines the object model, behavior, I/O communication and configuration data.
- CIP Statement of Compliance - Information specific to the objects implemented in the device. The Statement of Compliance identifies, for each object class and instance, the attributes, attribute access rules, services, and service options implemented in the device.

This information is required to construct a complete test for CIP conformance verification.

2.1.1 Application Network Terms

Definition of product functionality requires the use of network terms rather than individual product terms to accommodate both existing and future products.

The following network terms describe a product. These terms are defined in the CIP Specifications. They are listed here for reference:

- Client
- Consumer
- Group 2 Client
- Group 2 Only Client
- Group 2 Only Server
- Group 2 Server
- Originator
- Producer
- Server
- Target

Classifying a device using one or more of the preceding terms fully specifies device functionality as viewed from the network. The conformance test is not concerned with functionality that is not accessible from the network.

The specifics of device classification are best explained with an example. For instance, a Master device, as defined in the CIP Specification, DeviceNet Volume 3, would be classified as both a **Group 2 Client** and a **Group 2 Only Client**. From the given definitions, a Master device must perform the following functions:

- Take ownership of the Predefined Master/Slave Connection Sets within its **Group 2** and **Group 2 Only Servers**. This means that it must be able to communicate with these connection sets. Since the I/O connections within the Predefined Master/Slave Connection Set can be either Polled or Bit–Strobed, taking ownership of the Slave device implies that the Master must be able to Poll and/or Strobe its Slaves.
- Act as the UCMM port for its UCMM incapable Slaves.
- Screen for all explicit messages bound for the Slaves it “owns” and re-route those messages across the slaves Explicit Messaging connection.

In addition to performing these functions, a Master performs functions common to all devices, such as the Network Access function. Therefore defining a device using CIP network terms defines network visible functionality of the device.

2.1.2 Device Profile

Every CIP product should have a Device Profile. A Device Profile specifies the product object model, the exact format of I/O data, and the public interface to the product configuration data. This information fully defines device:

- behavior.
- I/O data communication.
- configuration and how that configuration affects behavior.

The Device Profile contains the information necessary to integrate the device into the CIP environment. To be in conformance with a particular Device Profile, the CIP device must have implemented all required objects specified by the Device Profile. In addition, the device may only have implemented objects contained in that Device Profile.

2.1.3 CIP Statement of Compliance

Since CIP products are modeled as a collection of objects, the CIP objects that are implemented in the DUT must be identified for the conformance test. The specific implementation of the optional attributes and services defined for each publicly accessible object must be specified. Given this data, a conformance test can be structured to test these specific objects. The test can confirm that object attributes and services specified in the product documentation have been implemented. In addition, a conformance test can identify objects, attributes, and services not included in the product documentation.

2.2 Test Identification

After device functionality is identified, the CIP conformance tests can be done. The following are the main CIP conformance tests and a brief description of each. These tests are defined in detail in chapters 3 and 4.

- **Network Access Test** - Tests that the device executes the Network Access State Machine (a.k.a. Duplicate MAC ID check) after resets and power cycles.
- **Transport Layer Test** - Tests the functionality of the Transport Layer, or fragmentation state machine, within the device under test.
- **Device Profile Verification** - Tests that object classes implemented in a device are those, and only those, listed in the Device Profile for the device.

- **Group 2 Server Poll/Bit Strobe Test** - Verifies that a Group 2 Server (or a Group 2 Only Server) correctly responds to the Poll and/or Bit Strobed I/O commands.
- **UCMM Connection Test** - Tests the existence of the UCMM port and verifies that server, client, or peer, devices perform the proper steps, in the proper sequence, when establishing Explicit Messaging connections.
- **Object Class Tests** - Tests support of attributes, services and adherence to specified behavior for the public object classes implemented in the device under test. Descriptions of object class tests are in chapter 4.

2.3 Test Reporting

Test reports shall provide documentary information including the date the conformance test is run, the vendor name and identification number, device catalog number, revision number, and profile name. For each object tested, the reports shall include the revision information for both the object tested as well as the revision information for the object specific Testware. Test reports may be presented in a Detailed or Summary format.

Reports shall state the pass/fail status and number of errors detected for each CIP technology test, object test and for the device. The reports shall include the pass/fail status and the number of errors detected for:

- each sub-section of a technology and object test.
- each technology and object test.
- the complete device test.

Any failure is sufficient to cause the general result for the device to be fail or non-conformant.

2.3.1 Detailed Reports

Each relevant CIP message sent and received during the test is included in the report. To enhance readability, redundant and unimportant messages are omitted. For example, access to attributes not defined in the CIP specifications are executed, but omitted from the report, unless errors are detected. The documentary and pass/fail information for the test is shown.

2.3.2 Summary Reports

The Summary report includes only the documentary and pass/fail information for the test. CIP messages are not recorded unless failures are detected. Summary information for any message that fails shall include both the request and response messages that caused the failure.

3. CIP Conformance Tests

This chapter defines format and content of the CIP Conformance Tests. The conformance tests are maintained as Network Technology tests and CIP Object tests. Network Technology Tests specify conformance tests for technology and communications functions defined in the Network Specification. These tests are defined in Chapter 4 of this document. CIP Object Tests specify conformance tests for objects defined in the CIP Specification. Network specific objects are defined in the network volumes. These tests are defined in Chapter 5 of this document.

3.0 Format and Content of CIP Conformance Tests

Each CIP Conformance Test contains the following sections:

- An introduction
- Functional Description
- Procedure Definition

The introduction precedes the Functional Description section and states the name of the test and when the test must be applied to a device. If necessary, other preliminary information is given.

The Functional Description section specifies the CIP functions or behavior the test verifies. The each function or behavior is listed with an identifying number.

The Procedure Definition section specifies how to test each function or behavior listed in the Functional Description section. Each of the listed items constitutes a step of the test. It contains subsections defining test initialization and message connection creation. The CIP messages needed to do the tests are given. When several CIP messages are needed to establish the behavior to be tested, the expected responses for these messages are assumed to be a Success_Response and are not included in the Procedure Description section.

The expected responses necessary to determine the Pass/Fail status of each step are specified for each step in the procedure. For tests intended to cause error responses, the expected general and additional error codes are defined. These responses must be observed. Otherwise, the object is classified non-conformant with the CIP Specification.

The exact protocol data units (PDUs) sent in the request or expected in the response are not presented. The contents of the PDUs are defined in the CIP Specifications. Sufficient detail is provided to allow the reader to compose requests and to interpret responses.

Tests shall demonstrate that objects conform to the CIP Specifications when using correct data for accessing attributes and invoking services. Tests shall also demonstrate that objects comply with the CIP Specifications when using incorrect data for accessing attributes and invoking services.

Whenever the Procedure Definition refers to a test timer, it indicates a timer under the control of the test. Timers implemented in the device are specifically referred to by the appropriate attribute.

When the Get_Attributes_All service is implemented, the default value specified in the CIP Specification is expected to be returned for any attributes not implemented in the object. If Vendor Specific attributes are implemented for an object, they are not covered by the conformance test.

Short examples of a Network Technology Test and CIP Object Tests are presented in the following sections with explanatory text. An example section explaining the use of Pass Tables in the conformance test descriptions is at the end of the chapter.

3.1 Example Technology Test

This section defines the Example Technology Test. This test is required for all devices.

Functional description

This test verifies that the device:

1) produces a Duplicate MAC ID request message after a reset.

2) remains in the Off-Line state after the first Duplicate MAC ID Request message is produced.

Test Procedure

- Initialization
  Log the Network technology test name and revision.
- Message Connection
  No Explicit Messaging or I/O Connections are to be in the Established state.

1) Send a type zero Reset request to the Identity Object, instance 1.

   Pass: the device correctly produces the first Duplicate MAC ID Request message.

2)

- Start a 1.1 second test timer.
- Send an Open_Request to the UCMM port.
- Send an Allocate request to the Group 2 Only Unconnected Explicit Message Port.
  Pass: No_Response to any messages.

Step 1 is to verify that a Duplicate MAC ID request message is produced after a reset request. Therefore, the test Procedure Definition for step 1 assumes that a Success_Response is received for the Reset request. The Pass/Fail decision depends on the production of a Duplicate MAC ID request message.

Step 2 is to verify that the device remains in the Off–Line state after the first Duplicate MAC ID Request message is produced. This requires a series of tasks and messages. Therefore, the test Procedure Definition for step 2 starts a timer local to the test, sends an Open Explicit Message Connection request and an Allocate Predefined Master/Slave Connection request to the device. The Pass/Fail decision depends on the test receiving no response to any messages.

3.2 Example Object Test

This section defines the Example Object Test. This test is required when the Example Object is implemented in the device.

Functional Description

This test verifies the:

1) class attribute access rules for the object.

2) instance attribute access rules for the object.

3) Common Service implementations for class and instance.

4) Object–specific and Reserved service implementations for class and instance.

5) conformance when incorrect data is used to access attributes and services.

6) object behavior accessible from the CIP network.

Procedure Definition

- Initialization
  The Example object must be available. Log the object name and object test revision. Describe any initialization required to properly set up the testing environment. This includes bringing the object to a state so that the object can be tested.
- Message Connection
  Describe Explicit Message or I/O connections needed to test the object.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined(00)	[Attribute_Not_Supported (x14)]
name of (01..99)	expected response for each attribute 1..99, one per row

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined(00)	[Attribute_Not_Supported (x14)]
name of (01..99)	expected response for each attribute 1..99, one per row

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (0x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined(00)	[Attribute_Not_Supported (x14)]
name of (01..99)	expected response for each attribute 1..99, one per row

- Request a Set_Attribute_Single service (x10) addressed to the a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined(00)	[Attribute_Not_Supported (x14)]
name of (01..99)	expected response for each attribute 1..99, one per row

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero, using message data shown. Reset service is tested in the Behavior Test.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Service_Name (01..49)	expected response for each attribute 1..49, one per row

- Request each of the Common Services, 00 - 49, addressed to Instance ID 1.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Service_Name (01..49)	expected response for each attribute 1..49, one per row

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	expected response for each attribute 1..49, one per row

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each of the Object–specific Services, 75 - 99, addressed to Instance ID 1.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	expected response for each attribute 1..49, one per row

- Request each of the Reserved Services, 100 - 255, addressed to Instance ID 1.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

- Define the steps needed to test the error conditions, for example

- For each attribute that has Set access, request a Set_Attribute_Single service that contains incorrect, insufficient or superfluous data. Show the expected responses for a Set_Attribute_Single error test.
  Pass: Error_Response, Invalid_Attribute_Value (x09)
- For each of the implemented Common and Object–specific Service, attempt to cause anomalous behavior by sending incorrect, insufficient or superfluous data to the object. Show the expected responses for a the error test.
  Pass: Error_Response, Invalid_Parameter (x20)

6) Behavior Tests

- Define the steps needed to test the behaviors
  Pass: The expected responses for each behavior test.

Behavior tests for CIP conformance are done only for behavior observable and accessible from the network. These are not functional behavior tests. Services that cause a significant change to the state of the device, i.e. Reset, are tested in a controlled fashion in the Behavior section of the test.

Where possible, objects are tested in states that can be achieved and observed from the network. If states of an object can be established from the network, i.e. external to the device in which the object resides, then those states are established and the resulting behavior observed for conformance to the CIP Specification. States that are defined for an object but are controlled internally by the device must be tested by the device developer.

When information in the Implementation Notes section of the Object Definition document applies to any behavior exhibited by the object that will be observable from the network, this section of the test definition describes tests to demonstrate correct implementation of the behavior.

3.3 Pass Tables:

Pass tables define expected responses for attribute access rules and service request tests. Information in the tables is intended to cover cases where attributes or services may or may not be implemented. Expected responses shown in the tables adhere to the following guidelines. Information needed to clarify responses shown in the tables shall be provided by the test definition.

A set of expected responses is enclosed in brackets, i.e. [A, B, C]. Any of the expected responses results in a pass for that test. Any other response results in a fail for that test. When it is known which expected response applies to a test, that response is the only one accepted for a pass result.

- If an object is not implemented, the expected response is Object_Does_Not_Exist (x16).
- If a service is not supported, the expected response is Service_Not_Supported (x08).
- If the requested service is supported but the specified attribute is not, the expected response is Attribute_Not_Supported (x14).
- If an attribute is implemented, Get_Attribute_Single must return a Success_Response.
- If the specified attribute is implemented with Set access, Set_Attribute_Single must return a Success_Response when used with valid data, and the object state allows the Set access.
- If the specified attribute is implemented with Set access, and the object state does not allow the Set access, the expected response is Object_State_Conflict (x0C).
- If the specified attribute is implemented with no Set access, Set_Attribute_Single must return an Error_Response with general error code Attribute_Not_Settable (x0E).

Expected values returned by Get_Attribute_Single or Get_Attributes_All responses are shown as TYPE (values), i.e. USINT(0..255) means that a one byte value of 0 to 255 is acceptable. Any other value will fail.

- A single value is shown as UINT(1). The value must be a two byte value = 1.
- A range of values is shown as USINT(1..10). The value must be a one byte value in the range 1 through 10 inclusive.
- A set of values is shown as BYTE(1, 3, 5, 9, 255); the value must be a one byte value shown in the set.

When expected values are shown i.e. USINT(0..3), the Success_Response is implied.

When expected values are shown with no value specification i.e. USINT, the full range of valid values for that type are accepted.

Array data is shown as TYPE[size]. For example, UINT[Max_value]. The expected result is an array of UINTs that may contain any value (0..65535). Array size is defined by the Max_value.

An array containing pairs of data is shown as (USINT, USINT)[size]. Each array element is a USINT value (0..255). size is a previously defined attribute value.

When a CIP general error code as an expected response, the complete Error_Response is expected i.e. Service_Not_Supported (x08) means 94 08 FF.The following table shows how expected responses to Get_Attribute_Single requests are presented.

Pass: The expected response for Get_Attribute_Single service from table below.

Attribute Name (ID)	**[Expected Values, Responses]**
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT(1), Attribute_Not_Supported]
Max_Instance (02)	[UINT]
Data_Type (03)	[USINT(1..3)]
Object_List (04)	[(UINT, UINT[size]), Attribute_Not_Supported] size is first UINT
Bound_Instances (05)	[USINT(0..255), Attribute_Not_Supported]
Binding (Class, Instance) List (06)	[(UNIT, UINT)[Bound_Instances], Attribute_Not_Supported]

The table shows that the attribute:

- Revision is optional and if implemented must have a two byte value = 1.
- Max_Instance is a required and may have any two byte value.
- Data_Type is required and must have a one byte value = 1, 2, or 3.
- Object_List is optional and if implemented must have a two byte size value 0..65535 and an array of two byte values specified by size.
- Bound_Instances is optional and if implemented must have a one byte value 0..255.
- Binding is optional and if implemented must have an array of two, two byte values 0..65535. The size of the array of two byte pairs is specified by Bound_Instances. In this example, since the array of pairs represents (Class, Instance), the test may specify further value limits for the class or instance i.e. (UINT(x08, x09), UINT(1..500)).

The following table shows how expected responses to Set_Attribute_Single requests are presented.

Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[Service_Not_Supported, Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Max_Instance (02)	[Success_Response, Service_Not_Supported, Attribute_Not_Settable]
Data_Type (03)	[Service_Not_Supported, Attribute_Not_Settable]
Object_List (04)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Bound_Instances (05)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Binding (06)	[Attribute_Not_Settable, Attribute_Not_Supported]

The table shows that Set_Attribute_Single is optional and the attribute:

- Revision is optional and if implemented is not settable.
- Max_Instance is a required and may or may not be settable.
- Data_Type is required and is not settable.
- Object_List is optional and if implemented, may or may not be settable.
- Bound_Instances is optional and if implemented is not settable.
- Binding is optional and if implemented is not settable.

The following table shows how expected responses to Common Service requests are presented.

Pass: The expected response for Common Services from the table below.

Service Name (Code)	[Expected Responses]
(00..04)	[Service_Not_Supported (x08)]
Reset (05)	[Success_Response]
(06..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response requested value]
(14..99)	[Service_Not_Supported]
(15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Service_Not_Supported]
(14..49)	[Service_Not_Supported]

The table shows that:

- Common Services 00..04 are not supported for the object.
- Common Services Reset is required.
- Common Services 06..13 are not supported for the object.
- Common Service Get_Attribute_Single is required
- Common Service 15 is not supported for the object.
- Common Service Set_Attribute_Single is optional.
- Common Services 17..49 are not supported for the object.

4. CIP Object Tests

This chapter specifies the conformance tests for CIP objects defined in the CIP Specification, Volume 1. A template for use in developing CIP Conformance test specifications is provided in Appendix A – CIP Object Test Definitions

CIP Network-specific and CIP Safety related tests are documented separately:

CIP Conformance Test Specification: DeviceNet Adaptation (ODVA PUB00167)

CIP Conformance Test Specification: EtherNet/IP Adaptation (ODVA PUB00168)

CIP Conformance Test Specification: CIP Safety Adaptation (ODVA PUB00170)

Object Code	For information about this Object Test:	Go to this page:
x01	Identity	30
x02	Message Router	37
x03	DeviceNet	PUB00167
x04	Assembly	41
x05	Connection	48
x06	Connection Manager	66
x07	Register	82
x08	Discrete Input Point	86
x09	Discrete Output Point	90
x0A	Analog Input Point	98
x0B	Analog Output Point	102
x0E	Presence Sensing	111
x0F	Parameter	115
x10	Parameter Group	123
x12	Group	128
x1D	Discrete Input Group	132
x1E	Discrete Output Group	136
x1F	Discrete Group	144
x20	Analog Input Group	148
x21	Analog Output Group	152
x22	Analog Group	160
x23	Position Sensor	164
x24	Position Controller Supervisor	168
x25	Position Controller	174
x26	Block Sequencer	182
x27	Command Block	186

x28	Motor Data	192
x29	Control Supervisor	198
x2A	AC/DC Drive	206
x2B	Acknowledge Handler	213
x2C	Overload	222
x2D	SoftStart	226
x2E	Selection	231
x30	S-Device Supervisor	238
x31	S-Analog Sensor	251
x32	S- Analog Actuator	263
x33	S-Single Stage Controller	270
x34	S-Gas Calibration	280
x35	Trip Point	285
x37	File	291
x39	Safety Supervisor	PUB00170
x3A	Safety Validator	PUB00170
x3B	Safety Discrete Output Point	PUB00170
x3C	Safety Discrete Output Group	PUB00170
x3D	Safety Discrete Input Point	PUB00170
x3E	Safety Discrete Input Group	PUB00170
x3F	Safety Dual Channel Output	PUB00170
x42	Motion Device Axis Object	322
x43	Time Sync	302
x47	Device Level Ring	PUB00168
x48	QoS	PUB00168
x49	Safety Analog Input Point	PUB00170
x4B	Safety Dual Channel Analog Input	PUB00170
x45	Originator Connection List	326
x4D	Target Connection List	329
x4E	Base Energy Object	309
x4F	Electrical Energy Object	314
x50	Non-Electrical Energy Object	318
x51	Base Switch Object	PUB00168
x53	Power Management Object	333
x54	RSTP Bridge Object	PUB00168
x55	RSTP Port Object	PUB00168
x56	PRP/HSR Protocol Object	PUB00168
x57	PRP/HSR Nodes Table Object	PUB00168
x5C	Power Curtailment Object	337

xF3	Connection Configuration	PUB00170
xF4	Port	305
xF5	TCP/IP Interface	PUB00168
xF6	Ethernet Link	PUB00168

4.0 Persistent Verifications for Protocol Conformance

The CIP protocol requires that messages have the specified format and content. For example, the Open Explicit Message Success Response must contain six bytes. The first byte is the Frag bit, XID bit and MACID. The second byte is the R/R bit and Service code, x4B. The third byte is two nibbles. The high order nibble must contain all zeros. The low order nibble specifies the Actual Message Body Format. The fourth byte is two nibbles. The high order nibble is the Destination Message ID. The low order nibble is the Source Message ID. The fifth and sixth bytes contain the Connection instance Id for the new connection. Whenever an Open Explicit Success Response is received, the data in the response is verified. The following list specifies CIP messaging verifications that are required but are not specifically stated with every occurrence of a message.

1. The Open Explicit Message Success_Response is validated as stated above.
2. The Connection instance Id is checked to verify that it is not one of the Ids reserved for supported Predefined Master/Slave Connection instances.
3. The Allocate Master/Slave Connection Set Success_Response contains three bytes. The first byte is the Frag bit, XID bit and MACID. The second byte is the R/R bit and Service code, x4B. The third byte is two nibbles. The high order nibble must contain all zeros. The low order nibble specifies the Actual Message Body Format.
4. DeviceNet Connection Identifiers contain the correct message group, MacId and Message Id.
5. Explicit Message message headers contain the correct MacId.
6. An Error_Response must contain correct number of bytes and the correct General Status/Error Code and Additional Status/Error Code.
7. DeviceNet explicit message response contains the correct XID bit value.
8. The Get_Attribute_Single Success_Response contains the correct number of bytes depending on the attribute data type.
9. Any response with reserved bits, the reserved bits contain the value zero.
10. The Set_Attribute_Single Success_Response for the Connection EPR attribute contains four bytes and the actual EPR value is as requested or rounded up.
11. The Set_Attribute_Single Success_Response for the Connection Production Inhibit Time attribute contains four bytes and the actual value is as requested or rounded up.
12. The Apply_Attribute Success_Response contains four bytes and the actual produced and consumed connection identifier values that are correct.
13. Device Shutdown message contains eight bytes, service code 0x4E and valid code.
14. General error code 0x2A, is accepted from a Group 2 Only Server instead of Service Not Supported, Attribute Not Settable or Attribute Not Supported codes.
15. The Set_Attribute_Single Success_Response has no data, except for timer attributes.

4.0.1 Object Class Attribute Access Tests

The class level attributes for all objects shall be tested according to the following procedures. Where necessary, the individual object tests shall specify exceptions to these rules.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT(1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT , Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

4.1 Identity Object Test x01

This section defines the Identity Object conformance test. This test is required for all devices.

Functional Description

This test verifies the:

1) class attribute access rules for the Identity object.

2) instance attribute access rules for the Identity object.

3) Common Service implementations for class and instance.

4) Object–specific and Reserved Service implementations for class and instance.

5) conformance when incorrect data is used to access attributes and services.

5.1) response when the Class Reset service is requested with invalid parameter data.

5.2) response when the Instance Reset service is requested with invalid parameter data.

5.3) response when the Get_Member service is requested with invalid parameter data.

5.4) response when the Semaphore attribute is set with invalid parameter data.

6) object behavior accessible from the network.

6.1) response when the Class Reset service is requested with valid parameter data.

6.2) response when the Instance Reset service is requested with valid parameter data.

6.3) implementation of Find_Next_Object_Instance.

6.4) implementation of Device Heartbeat. (for DeviceNet only)

6.5) implementation of Class attrbute Max Instance and multiple instances.

6.6) implementation of Get_Member service and International String attributes.

6.7) implementation of the Semaphore attribute.

Procedure Definition

- Initialization
  The Identity object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT , Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT (1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT (7..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access

attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (0x16) indicates an instance is found. Use found instance ID for any tests that require an instance.
Pass: The Identity object must have instance 1 implemented

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Vendor (01)	[UINT (1..65535)]
Device_Type (02)	[UINT]
Product_Code (03)	[UINT (1..65535)]
Revision (04)	[struct(USINT (1..127), USINT (1..255))]
Status (05)	[WORD(0000xxxxxxxx0x0x)] 0 = must be zero, x = 0 or 1
Serial_Number (06)	[(UDINT (0..4294967295)]
Product_Name (07)	[SHORT_STRING] Maximum 32 Characters
State (08)	[USINT(1..5), Attribute_Not_Supported]
Configuration_Consistency_Value(09)	[UINT, Attribute_Not_Supported]
HeartBeat_Interval (10)	[USINT(0..255), Attribute_Not_Supported]
Active_Language (11)	[(USINT, USINT, USINT), Attribute_Not_Supported]
Supported_Language_List (12)	[[(USINT, USINT, USINT)], Attribute_Not_Supported]
International_Product_Name (13)	[STRINGI, Attribute_Not_Supported]
Semaphore (14)	[UINT(0), UDINT(0), ITIME (0), Attribute_Not_Supported]
Assigned_Name (15)	[STRINGI, Attribute_Not_Supported]
Assigned_Description (16)	[STRINGI, Attribute_Not_Supported]
Geographic_Location (17)	[STRINGI, Attribute_Not_Supported]
Protection Mode (19)	[WORD, , Attribute_Not_Supported]
Uptime (20)	[UDINT, Attribute_Not_Supported]
Undefined (21..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Vendor (01)	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
Device_Type (02)	[Service_Not_Supported, Attribute_Not_Settable]

Attribute Name (ID)	[Expected Responses]
Product_Code (03)	[Service_Not_Supported, Attribute_Not_Settable]
Revision (04)	[Service_Not_Supported, Attribute_Not_Settable]
Status (05)	[Service_Not_Supported, Attribute_Not_Settable]
Serial_Number (06)	[Service_Not_Supported, Attribute_Not_Settable]
Product_Name (07)	[Service_Not_Supported, Attribute_Not_Settable]
State (08)	[Service_Not_Supported, Attribute_Not_Supported, Attribute_Not_Settable]
Configuration_Consistency_Value (09)	[Service_Not_Supported, Attribute_Not_Supported, Attribute_Not_Settable]
HeartBeat_Interval (10)	[Success_Response, Attribute_Not_Supported]
Active_Language (11)	[Success_Response, Service_Not_Supported, Attribute_Not_Supported]
Supported_Language_List (12)	[Service_Not_Supported, Attribute_Not_Supported, Attribute_Not_Settable]
International_Product_Name (13)	[Service_Not_Supported, Attribute_Not_Supported, Attribute_Not_Settable]
Semaphore (14)	[Success_Response, Attribute_Not_Supported]
Assigned_Name (15)	[Attribute_Not_Supported, Attribute_Not_Settable]
Assigned_Description (16)	[Attribute_Not_Supported, Attribute_Not_Settable]
Geographic_Location (17)	[Attribute_Not_Supported, Attribute_Not_Settable]
Protection_Mode (19)	[Attribute_Not_Supported, Attribute_Not_Settable]
Uptime (20)	[Attribute_Not_Supported, Attribute_Not_Settable]
Undefined (21..99)	[Service_Not_Supported, Attribute_Not_Supported]

3) Common Services Test

3.1) Class Implementation

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero, using message data shown. Reset service is tested in the Behavior Test. Use Instance ID 0, Max_Values 255 for testing the Find_Next_Object_Instance.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]0
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..04)	[Service_Not_Supported]
Reset (05)	[Success_Response, Service_Not_Supported] Note 1
(06..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response, Service_Not_Supported, if no attributes are supported]
(15..16)	[Service_Not_Supported]
Find_Next_Object_Instance (17)	[(UINT(1..255), UINT[size]), Service_Not_Supported] size = first UINT
Reserved (19..49)	[Service_Not_Supported]

Note 1: For Safety Devices, See CIP Safety PCTS (ODVA PUB00170) for test procedures

3.2) Instance Implementation

- Request each of the Common Services, 00 - 49, addressed to Instance ID 1.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]

Service Name (Code)	[Expected Responses]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..04)	[Service_Not_Supported]
Reset (05)	[Success_Response] then network access state machine. Note 1
(06..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15..17)	[Service_Not_Supported]
Get_Member (18)	[Success_Response, if STRINGI attributes are implemented Service_Not_Supported]
Reserved (19..49)	[Service_Not_Supported]

Note 1: For Safety Devices, See CIP Safety PCTS (ODVA PUB00170) for test procedures

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each of the Object–specific Services, 76 - 99, addressed to Instance ID 1.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (code)	[Expected Responses]
Object–specific Codes (76..99)	[Service_Not_Supported (x08)]

- Request each of the Reserved Services, 100 - 255, addressed to Instance ID 1.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) Request a Reset service addressed to the class using data shown.
Reset, Instance ID 0, Type 2
Reset, Instance ID 0, Type 255
Pass: The expected response for Reset request from the table below.

Service Name (code)	Implemented in Class	NOT Implemented in Class
Reset (05), Type 2	[Invalid_Parameter (x20)]	[Service_Not_Supported (x08)]
Reset (05), Type 255	[Invalid_Parameter]	[Service_Not_Supported]

5.2) Request a Reset service addressed to the class using data shown.
Reset, Instance ID 1, Type 2
Reset, Instance ID 1, Type 255

Pass: The expected response for Reset request from the table below.

Service Name (code)	[Expected Responses]
Reset (05), type 2	[Invalid_Parameter (x20)]
Reset (05), type 255	[Invalid_Parameter]

5.3) Service Test, if implemented.

- Request a Get_Member, Destination_List, MemberId = 255.
  Pass: Error_Response, 94 20 FF, Invalid Parameter

5.4) Semaphore Error Test, if implemented.

- Request a Set Attribute with Too Much Data.
  Pass: Error_Response, 94 15 FF, Too Much Data
- Request a Set Attribute with Not Enough Data.
  Pass: Error_Response, 94 13 FF, Not Enough Data
- Request a Set Attribute time = 0..99, 0x8000.
  Pass: Error_Response, 94 09 FF, Invalid Attribute Value

6) Behavior Tests

6.1) Request a Reset service addressed to the class using data shown. The device must execute the network access procedure, Duplicate MAC ID check, after the reset.
Reset, Instance ID 0, Type 0
Reset, Instance ID 0, Type 1
Reset, Instance ID 0 (Default)

Pass: The expected responses for Reset request to the Class from the table below.

Service Name (code)	Implemented in Class	NOT Implemented in Class
Reset (05), Type 0	[Success_Response + Network Access]	[Service_Not_Supported (x08)]
Reset (05), Type 1	[Success_Response + Network Access, Invalid_Parameter (x20)]	[Service_Not_Supported]
Reset (05), Default	[Success_Response + Network Access]	[Service_Not_Supported]

- If a Success_Response is returned, request a Get_Attribute_Single service, Status attribute.
  Pass: Success_Response with a valid Status attribute (no reserved bits set).

6.2) Request a Reset service addressed to each instance using data shown. The device must execute the network access procedure, Duplicate MAC ID check, after the reset.
Reset, Instance ID x, Type 0
Reset, Instance ID x, Type 1
Reset, Instance ID x (Default)

Pass: The expected responses for Reset request to the Instance from the table below.

Service Name (code)	[Expected Responses]
Reset (05), Type 0	[Success_Response + Network Access]
Reset (05), Type 1	[Success_Response + Network Access, Invalid_Parameter_Value (0x20)]
Reset (05), Default	[Success_Response + Network Access]

Note: Return Error_Response 94 10 ff, Device State Conflict, if unable to accept request.

- If a Success_Response is returned, request a Get_Attribute_Single service, Status attribute.
  Pass: Success_Response with a valid Status attribute (no reserved bits set).
- After a Type 0 Reset, check if the device still accepts EM request 3 seconds after the successful Reset response.

  Pass: The DUT stops responding EM request 3 seconds after sending Reset success.

6.3) If Find_Next_Object_Instance is implemented for the class,

- Request a Find_Next_Object_Instance, Instance ID = 0, Max_Return_Values = 255.
  Pass: Success_Response, first USINT = size, Array of UINT[size].
- Request a Find_Next_Object_Instance, Instance ID = Array[size], Max_Return_Values 255.
  Pass: Success_Response, USINT = 0.
- Request a Find_Next_Object_Instance, Instance ID = 0, Max_Return_Values = 1.
  Pass: Success_Response, first USINT = 1, Array[1] = 1.

6.4) Device HeartBeat Test. Omitted if HeartBeat Interval is not implemented.

- Request a Set_Attribute_Single, Instance ID = 1, HeartBeat_Interval = 2.
  Pass: Success_Response
- Start a test timer to expire in 2 seconds, when the timer expires, check the input message queue.
  Pass: Device Heartbeat Message received. (Identifier is correct)
- Check the Device Heartbeat Message format
  Pass: MacId = DUT MacId, R/R bit = 1, Service Code = x4D, Instance > 0, byte 5, bits 3-7 = 0.
- Start a test timer to expire in 2 seconds, when the timer expires, check the input message queue.
  Pass: Device Heartbeat Message received.
- Start a test timer to expire in 2 seconds
- Set the Explicit Message EPR = 100 milli-seconds and allow the connection to time out.
- When the 2 second test timer expires, check the input message queue.
  Pass: Device Heartbeat Message received. Correct format, SF bit (2) = 1.
- Request a Set_Attribute_Single, Instance ID = 1, HeartBeat_Interval = 0.
  Pass: Success_Response
- Start a test timer to expire in 2 seconds, when the timer expires, check the input message queue.
  Pass: No_Response. No Device Heartbeat Message received.

6.5) Check Multiple Insances.

- Request a Get_Attribute_Single, Class, Max_Instance.
  Pass: Success_Response
- For each instance between the first one found to Max_Instance, request a Get_Attribute_Single, instance, attribute 1.
  Pass: Success_Response
- Request a Get_Attribute_Single, Max_Instance + 1, attribute 1.
  Pass: Error_Response, 94 16 ff, Object Does Not Exist
- For scattered multiple instances, verify Find_Next_Object_Instance is implemented.

  Pass: Success_Response for implemented instances

6.6) Check International Strings.

- request a Get_Attribute_Single, Active_Language.
  Pass: Success_Response, length = 3 bytes, valid language identifier
- request a Get_Attribute_Single, Supported_Language_List.
  Pass: Success_Response
- validate each language identifier, determine the number of identifiers
- requestGet_Member, International_Product_Name, each member of Supported_Language_List.
  Pass: Success_Response and valid STRINGI data for requested language
  Pass: Error_Response, 94 02 FF, Resource_Unavailable

6.7) Semaphore Behavior Test,

- Request a Get Attribute Single, Semaphore.
  Pass: Success_Response, 0, 0, 0
- Request a Set Attribute Single, value = 1, 1, 100.
  Pass: Success_Response.
- Request a Set Attribute Single, value = 2, 1, 100.
  Pass: Error_Response, 94 02 FF, Resource_Unavailable
- Request a Set Attribute Single, value = 1, 2, 100.
  Pass: Error_Response, 94 02 FF, Resource_Unavailable
- Request a Get Attribute Single, Semaphore.
  Pass: Success_Response, 1, 2, timer value
- Start a test timer value = retuned value of Semaphore timer.
- When timer expires, request a Get Attribute Single, Semaphore.
  Pass: Success_Response, 0, 0, 0

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

6.8) Flash_LEDs Service Test if it's implemented

- Value 1- 8 are invalid values to Time parameter of the Flash_LEDs service.
- The DUT needs to do 6 "red-green-off" sequences in 9 seconds.
- Value 0 of Time parameter stops the flashing.

4.2 Message Router Object Test x02

This section defines the Message Router object conformance test. This test is required for all devices.

Functional Description

This test verifies the:

1) Class attributes access rules for the Message Router object.

2) Instance attributes access rules for the Message Router object.

3) Common Service implementations for class and instance.

4) Object–specific and Reserved service implementations for class and instance.

5) Conformance when incorrect data is used to access attributes and services.

6) Object behavior accessible from the network.

Procedure Definition

- Initialization
  The Message Router object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT (1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT (0..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Object_List (01)	[(UINT (1..1223), UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Number_Available (02)	[UINT , Service_Not_Supported, Attribute_Not_Supported]
Number_Active (03)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Active_Connections (04)	[UINT[Number_Active], Service_Not_Supported, Attribute_Not_Supported]
Undefined (05..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08), Attribute_Not_Settable (x0E), Attribute_Not_Supported (x14)]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response] [Service_Not_Supported] if no attributes are supported
Reserved (15..49)	[Service_Not_Supported]

Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..09)	[Service_Not_Supported]
Multiple_Service_Packet (10)	[Not_Enough_Data] – see **note 1** below
(11..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response] [Service_Not_Supported] if no attributes are supported
(15..49)	[Service_Not_Supported]

Note 1: a manual test procedure is required for validation of this service. Currently, the

conformance test logs this requirement as an error if the service is implemented.

4) Object–specific and Reserved Services Test

- Request each Object–specific services, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
SymbolicTranslation (75)	[Success_Response, Service_Not_Supported (x08)]
Object–specific Codes (76..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific services, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each of the Reserved Services, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests

The MR.2.E Request Path Error Tests modify bytes in a valid request path for a Forward Open request to make it invalid.

MR.2.E Request Path Size too small

connection path size = connection path size + 1

Pass: NOT_ENOUGH_DATA, PATH_SIZE_INVALID, PATH_SEG_ERROR, PATH_DEST_UNK

MR.2.E Wrong Class Number: Message_Router_Request.Path[1] = 0

Pass: Object does not exist (0x16) or PATH_DEST_UNK

MR.2.E Wrong Instance (logical number): Message_Router_Request.Path[3] = 0

Pass: Object does not exist (0x16) or PATH_DEST_UNK

MR.2.E Wrong Class Segment: Message_Router::RequestPath[0] = 0xFF
Pass: Path Segment Error (0x4)

MR.2.E Wrong Logical Segment: Message_Router::RequestPath[2] = 0xFF
Pass: Path Segment Error (0x4)

6) Behavior Tests.
NOTE: Message Router Behavior is tested indirectly throughout the conformance tests.

6.1) Multiple Service Packet

This is a Common Service [Vol 1 Ed 3.9 A-4.10].

NOTE: a manual test is required if implemented on DeviceNet. Please contact ODVA.

6.1.1) Single Service Request

- Request a Multiple_Service_Packet service (x0A) addressed to instance 1 of the Message Router object containing a single request Get_Attribute_Single request for Identity instance attribute 1.
  Pass: .successful Multiple Packet Service response with Vendor ID data

6.1.2) Multiple Service Request

- Request a Multiple_Service_Packet service (x0A) addressed to instance 1 of the Message Router object containing four requests to Get_Attribute_Single Identity instance attributes 1 through 4.
  Pass: .successful Multiple Packet Service response containing four successful responses with Identity instance attribute data for Vendor ID, Device Type, Producet Code, and Revision.

6.1.3) Multiple Service Request With Error

- Request a Multiple_Service_Packet service (x0A) addressed to instance 1 of the Message Router object containing three requests to Get_Attribute_Single Identity instance attributes 1 through 3.and one request to Get_Attribute_Single Message Router instance attribute 5 (should not exist)
  Pass: .error status response 0x1e (Embedded service error) and Multiple Packet Service response (containing three successful responses with Identity instance attribute data for Vendor ID, Device Type, Producet Code, and one error response 0x14 Attribute not supported.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.3 Assembly Object Test x04

This section defines the Assembly object conformance test. This test is required when the Assembly object is implemented in the device.

Functional Description

This test verifies the:

1) Attribute access rules for the Assembly class object.
2) Service implementations for the class.
3) Attribute access rules for each Assembly instance.
4) Service implementations for each instance.
5) Conformance when incorrect data is used to access attributes and services.
5.1) Response to Set_Attribute_Single, Data, data size invalid.
6) Object behavior accessible from the network
6.1) Behavior of the Assembly when Data attribute is Settable.
6.2) Creation of an Assembly Instance.
6.3) Insert_Member Service behavior.
6.4) Set_Attribute_Single Service behavior.
6.5) Remove_Member Service behavior.
7) Conformance when incorrect data is used to access attributes and service for Dynamic Assemblies.
7.1) Insert_Member, Member_Location is invalid.
7.2) Insert_Member, Member path size is too big or too small.
7.3) Remove_Member, Member_Location is invalid.
8.1) Remove_Member, all members
8.2) Delete request for a Dynamic Assembly Instance

Procedure Definition

- Initialization
  The Assembly object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (2), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT , Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT (1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT (1..199), Service_Not_Supported, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Num_Members (01)	[UINT] If Dynamic Assembly [UINT (1..65535), Attribute_Not_Supported] If Static Assembly
Member_List (02)	[Array_Member[Num_Members]] If Dynamic Assembly [Array_Member[Num_Members], Attribute_Not_Supported] If Static Assembly
Array_Member	
Member_Description	[INT (1..32767)] data size in bits, bit 15 is direction bit, 1 = write
Member_Path_Size in bytes	[UINT (6..65535)]
Member_Path	EPATH [Member_Path_Size] see Note 1 below
Data (03)	[Product Specific] If Member_List is implemented, the Member Data Description can be used to interpret the data. Otherwise, this information must be Supplied by the vendor.
Size	[UINT, Attribute_Not_Supported]
Undefined (05..99)	[Attribute_Not_Supported]

Note 1: EPATH encoding is defined in the CIP Specification, Volume 1, Appendix C, Abstract Syntax Encoding for Segment Types. The expected values for a EPATH are as follows.
Segment Type (bits 7, 6, 5) - 1, Logical Segment.
Logical Format (bits 4, 3, 2) - 0, 1, 4; Class ID, Instance ID, Attribute ID.
Logical Data Format (bits 1, 0) - 0, 1; 0 = USINT(1..255), 1 = UINT(1..65535).
An EPATH must specify Class ID, and Instance ID. Attribute ID is optional.

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses] Static Assembly
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]

Num_Members (01)	[Service_Not_Supported, Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Member_List (02)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Data (03)	[Success_Response, Attribute_Not_Settable, Service_Not_Supported]
Size (04)	[Service_Not_Supported, Attribute_Not_Settable]
Undefined (05..99)	[Service_Not_Supported, Attribute_Not_Supported]

Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses] Dynamic Assembly
Undefined (00)	[Attribute_Not_Supported (x14)]
Num_Members (01)	[Attribute_Not_Settable (x0E)]
Member_List (02)	[Attribute_Not_Settable] Add_Member Implemented [Success_Response] Add_Member Not Implemented
Data (03)	[Success_Response]
Undefined (04..99)	[Attribute_Not_Supported]

Note: If Member_List is implemented, the Member Data Description can be used to interpret the Data array. Otherwise, this information must be Supplied by the vendor.

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.

Note: Delete the instance ID returned by the Create Service.

Pass: The expected response from table below for each Common Services request.

Service Name (Code)	[Expected Responses]
(00..07)	[Service_Not_Supported (x08)]
Create (08)	[UINT (1..65535)] Dynamic Assembly [Service_Not_Supported] Static Assembly
Delete (09)	[Service_Not_Supported] Static Assembly
(10..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] if no attributes are supported
(15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Services request.

Service Name (Code)	[Expected Responses]
(00..08)	[Service_Not_Supported (x08)]
Delete (09)	For Dynamic Assembly [Success_Response] For Static Assembly [Service_Not_Supported]
(10..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	For Dynamic Assembly [Success_Response] For Static Assembly [Service_Not_Supported]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- For Assemblies that do not support the Add_Member and Delete_Member services, request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- For Dynamic Assemblies that support the Add_Member and Delete_Member services, request each Object–specific Service, 77 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (77..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) Set Data with Invalid sizes. This test applies if the Data attribute is settable.

- Request a Set_Attribute_Single, Data, size of data too small.
  Pass: Success_Response, or Error_Response = 94 13 00, Not_Enough_Data
- Request a Set_Attribute_Single, Data, size of data too large.
  Pass: Error_Response = 94 15 00 , Too_Much_Data

6) Behavior tests.

6.0) Profile Verification Tests

Each Assembly Instance is verified according to the definition of that instance from the Device Profile. Each Assembly Instance must adhere to the Device Profile definiton for:

- Instance Id, or must be Vendor Specific Id.
- The correct type, input, output, or configuration.

- The correct size, in bytes, for that Instance Id.
- Required instances must be implemented.

6.1) If the Data attribute is settable and Member_List is implemented, verify set Data behavior.

- Request Get_Attribute_Single, Data.
- Modify the data for each member of the assembly.
- Request Set_Attribute_Single, Data = modified Data.
- Request a Get_Attribute_Single for each attribute in the Member_List.
  Pass: Each attribute has correct value from the Assembly object after Setting the Data attribute.

6.2) Check Size/Data

- Request a Get_Attribute_Single for Size attribute (if implemented)
- Compare the value for the Size attribute to the actual size of the Data attribute
  Pass: value for the Size attribute = actual size of the Data attribute.

The Dynamic Assembly tests are not implemented

These tests apply to Dynamic Assemblies. The test must be configured with data sufficient to create an Assembly instance and add the required attribute data. The test creates an Assembly using the instances of I/O objects, such as the Discrete Input or Discrete Output objects, implemented in the device. The test requires the object code(s), and attribute IDs of the input and output data. Steps 6.2 through 8.0 are done for both an input object and an output object. This data is specified in the CIP Statement of Compliance Typical I/O Data Address Path.

Note: The test for Add_Member behavior creates the Member_Path using only Logical segment type. Member_Path_Size is calculated based on values for class Id, instance Id, and attribute Id of the input or output object used from the Typical I/O Data Address Path. The Member_Data_Description value is calculated as (attribute size in bytes * 8) with bit 15 = 1 if the value is written to by the application object.

Note: If a Create request returns Service_Not_Supported, the Assembly is a Static Assembly.

Note: Steps 6.2 through 8.2 are done once for an Input Assembly and once for an Output Assembly. The Input Assembly is created using the Input object Id and attribute Id of the Typical I/O Data Address Path. The Output Assembly is created using the Output object Id and attribute Id of the Typical I/O Data Address Path.

6.3) Request a Create service.

Pass: Service Success and the created instance number.

6.4) Add_Member Service behavior

- Get_Attribute_Single, Num_Members.
  Pass: Success_Response, value = 0
- Get_Attribute_Single, Member_List.
  Pass: Success_Response, value = NULL
- Get_Attribute_Single, Data array.
  Pass: Success_Response, value = NULL

If Add_Member is not implemented, use a Set_Attribute_Single to set the Member_List.

- Instance_Id = Locate an instance of the input/output object class.
- Request an Add_Member service using;
  Member_Location = 0

Member_Data_Description = calculated size of input/output data
Member_Path_Size = calculated size of Member_Path, bit 15 = 1 if output object
Member_Path = Logical Segment, input/output attribute Id segment address.
Member_Description = Member_Data_Description, Member_Path_Size, Member_Path
Pass: Success_Response

- Request an Add_Member service using;
  Member_Location = 0
  Member_Description = as defined above for input/output object class.
  Pass: Success_Response
- Get_Attribute_Single, Num_Members.
  Pass: Num_Members = 2
- Get_Attribute_Single, Member_List.
  Pass: Member_List = New Array_Member data for two member elements
- Get_Attribute_Single, Data.
  Pass: Success_Response, Data size = Data is the input/output object attribute data size times 2.

6.5)

- Request Get_Attribute_Single, Assembly Instance, Data attribute.
  Pass: Success_Response and Assembly data for 2 members
- Request Set_Attribute_Single, Assembly Instance, Data attribute = previous data.
  Pass: The expected response as shown in the table below.

Output Object Instance	Input Object Instance
Success_Response	Error_Response, 94 0E FF, Attribute_Not_Settable

6.6) Omit this step if Delete_Member is not implemented.

- Request Delete_Member Service, Member = 1
  Pass: Service Success
- Request Get_Attribute_Single, Num_Members
  Pass: Num_Members = 1
- Get_Attribute_Single, Member_List.
  Pass: Member_List = New Array_Member data for one member element.
- Get_Attribute_Single, Data.
  Pass: Success_Response, Data size = Data is the input/output object attribute data size.

7.0) Error tests for Add_Member service. Use Assembly created in step 6.2

7.1) Invalid Member_Location test.

- Request an Add_Member service using;
  Member_Location = Num_Members + 2
  Member_Description = Valid member data for the input/output assembly attribute data.
  Pass: Error_Response = 94 20 FF , Invalid_Parameter

7.2) Invalid Member path size, too big or too small.

- Using Member_Description from step 7.1, add 1 to the Member_Path_Size.

- Increase path size for Member_Description for Member element data.
- Request an Add_Member service using;
  Member_Location = 0
  Member_Path_Size = the revised copy of Member_Description.
  Pass: Error_Response = 94 20 FF , Invalid_Parameter
- Using Member_Description from step 7.1, subtract 1 from the Member_Path_Size.
- Request an Add_Member service using;
  Member_Location = 0
  Member_Description = the revised copy of Member_Description.
  Pass: Error_Response = 94 20 FF , Invalid_Parameter

7.3) Request Delete_Member, Member_Location = Num_Members + 1.

Pass: Error_Response = 94 20 FF , Invalid_Parameter

8.1) If Delete_Member is implemented, Request Delete_Member, instance = 0.
Otherwise, request Set_Attribute_Single Member_List = NULL.

Pass: Success_Response

- Get_Attribute_Single, Num_Members
  Pass: Success_Response, value = 0
- Get_Attribute_Single, Member_List.
  Pass: Success_Response, value = NULL
- Get_Attribute_Single, Data array.
  Pass: Success_Response, value = NULL

8.2) Request Delete Service using Instance created in step 6.2

Pass: Success_Response

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.4 Connection Object Test x05

This section defines the Connection object conformance test. This test is required for all devices.

This test has three parts. Part one verifies Explicit Message connection behavior. Part two verifies I/O connection behavior. In part two, I/O connections are tested in each possible state. Part three verifies the class level attribute and service implementations.

Note: I/O connections are accessed by an Explicit Message connection. Explicit Message connections are accessed via themselves i.e., the getting and setting of the tested Explicit Message connection attributes is accomplished via that same Explicit Messaging connection.

Note: Detailed testing for configurable I/O connections requires specific information on how the connections are implemented. These tests will be added in a future revision of the test specification.

Functional Description

Part 1: Explicit Messaging Connection

State: **Non-Existent**

No meaningful testing is possible for a non–existent connection.

State: **Established**

This test verifies the:

1.1) attribute access rules when the connection instance is in the Established state.

1.2) error response when the Apply_Attributes service is requested.

1.3) Common Services implementation.

1.4) Object–specific and Reserved service implementation.

1.5) Delete request is processed properly.

1.6) Pre-consumption and Inactivity TimeOut behavior.

1.7) Deferred Delete behavior.

Part 2: I/O Connection

State: **Non-Existent**

This test verifies the:

2.1) error response when the Delete service is requested.

2.2) attribute access rules when the connection instance is in the non–existent state.

2.3) error response when the Reset service is requested.

2.4) error response when the Apply_Attributes service is requested.

State: **Configuring**

This test verifies the:

2.5) response to a Delete request.

2.6) instance attribute access rules.

2.6.1) Production Inhibit Timer rounding.

2.7) conformance when incorrect data is used to access attributes and services.

2.8) error response when the Reset service is requested.

2.9) connection transitions to the proper state when the Apply_Attributes service is requested.

State: **Waiting_for_Connection_ID**

This test verifies the:

2.10) response to a Delete request.

2.11) instance attribute access rules.
2.12) conformance when incorrect data is used to access attributes and services
2.13) error response when the Reset service is requested.
2.14) connection transitions to the proper state when the Apply_Attributes service is requested.

State: **Established**

This test verifies the:

2.15) response to a Delete request.
2.16) instance attribute access rules.
2.17) error response when the Apply_Attributes service is requested.
2.18) inactivity timer functions correctly.
2.19) inactivity timer is restarted when the Reset service is performed.
2.20) conformance when incorrect data is used to access attributes and services
2.21) transition to the proper state when the inactivity timer expires.

State: **Timed_Out**

This test verifies the:

2.22) instance attribute access rules.
2.23) error response when the Apply_Attributes service is requested.
2.24) conformance when incorrect data is used to access attributes and services
2.25) response when the Delete service is requested.
2.26) connection transitions to WatchDog_Timeout_Action state after time–out.

Part *3:* Class level testing

This test verifies the:

3.1) EPR time out behavior.
3.2) EPR timer rounding.
3.3) maximum EPR value time–out.
3.4) class Reset service implementation.
3.5) class attribute access rules.
3.6) Common Service implementations.
3.7) Object–specific and Reserved service implementations.
3.8) class Delete service implementation.

Test Procedure

- Initialization
  The Connection Object must be available. Log the object name and object test revision.
- Message Connection
  No Explicit Messaging or I/O Connections are to be in the Established state.

Part 1: Explicit Messaging Connection

State: **Non-Existent**

No meaningful testing is possible for a non–existent connection.

State: **Established**

1.1)

- Open/Allocate an Explicit Messaging (EM) connection. Use the following arguments when creating a connection via the UCMM: Message_Body_Format = 0, Group_Select = a supported group, Message Id as appropriate for the chosen Group_Select.
- Request a Set_Attribute_Single, Expected_Packet_Rate, value = zero.
- Request a Get_Attribute_Single, State.
  Pass: State = Established
- Request a Get_Attribute_Single service (x0E) addressed to the EM connection instance to access attributes 00 - 99. Save the returned value of each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	**[Expected Values, Responses]**
Undefined (00)	[Attribute_Not_Supported (x14)]
State (01)	[USINT (3)]
Instance_Type (02)	[USINT (0)]
TransportClass_Trigger (03)	[USINT (0..255)]
Produced_Connection_Id (04)	[UINT (x0000..x07F0)]
Consumed_Connection_Id (05)	[UINT (x0000..x07F0)]
Initial_Comm_Characteristics (06)	[USINT Two nibbles each of value 0..3]
Produced_Connection_Size (07)	[UINT]
Consumed_Connection_Size (08)	[UINT]
Expected_Packet_Rate (09)	[UINT (2500 or greater depending on timer tick value)]
Reserved (10, 11)	[Attribute_Not_Supported]
Watchdog_Timeout_Action (12)	[USINT (1)]
Produced_Connection_Path_Length (13)	[UINT]
Produced_Connection_Path (14)	[EPATH]
Consumed_Connection_Path_Length(15)	[UINT]
Consumed_Connection_Path (16)	[EPATH]
Production_Inhibit_Time (17)	[UINT (0), Attribute_Not_Supported]
Connection_Timeout_Multiplier (18)	[USINT (0), Attribute_Not_Supported]
Connection_Binding_List (19)	[UINT(size),UINT[size], Attribute_Not_Supported]
Undefined (20..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the EM connection instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
State (01)	[Attribute_Not_Settable (x0E), Service_Not_Supported]
Instance_Type (02)	[Attribute_Not_Settable, Service_Not_Supported]
TransportClass_Trigger (03)	[Attribute_Not_Settable, Service_Not_Supported]
Produced_Connection_Id (04)	[Attribute_Not_Settable, Service_Not_Supported]
Consumed_Connection_Id (05)	[Attribute_Not_Settable, Service_Not_Supported]
Initial_Comm_Characteristics (06)	[Attribute_Not_Settable, Service_Not_Supported]
Produced_Connection_Size (07)	[Attribute_Not_Settable, Service_Not_Supported]
Consumed_Connection_Size (08)	[Attribute_Not_Settable, Service_Not_Supported]
Expected_Packet_Rate (09)	[Success_Response with EPR value (UINT), Service_Not_Supported]
Reserved (10, 11)	[Attribute_Not_Supported, Service_Not_Supported]
Watchdog_Timeout_Action (12)	[Success_Response, Attribute_Not_Settable, Service_Not_Supported]
Produced_Connection_Path_Length (13)	[Attribute_Not_Settable, Service_Not_Supported]
Produced_Connection_Path (14)	[Attribute_Not_Settable, Service_Not_Supported]
Consumed_Connection_Path_Length(15)	[Attribute_Not_Settable, Service_Not_Supported]
Consumed_Connection_Path (16)	[Attribute_Not_Settable, Service_Not_Supported]
Production_Inhibit_Time (17)	[Attribute_Not_Settable, Service_Not_Supported]
Connection_Timeout_Multiplier (18)	[Success_Response, Service_Not_Supported]
Connection_Binding_List (19)	[UINT(size),UINT[size], Attribute_Not_Supported]
All other public attributes (20..99)	[Attribute_Not_Supported, Service_Not_Supported]

1.2) Apply an Established Explicit Msg Connection

- Send an Apply_Attributes request to the connection.
  Pass: Error_Response: 94 0C FF, Object_State_Conflict or 94 08 FF, Service_Not_Supported

1.3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the EM connection Instance.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
(0..4)	[Service_Not_Supported (x08)]
Reset (05)	[Success_Response, Service_Not_Supported]
(6..8)	[Service_Not_Supported]
Delete (9)	Not Tested Here
(10..12)	[Service_Not_Supported]
Apply_Attributes (13)	Not Tested Here
Get_Attribute_Single (14)	[Success_Response]
(15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Service_Not_Supported, Attribute_Not_Settable (x0E)]
(17..49)	[Service_Not_Supported]

1.4) Object–specific and Reserved Services Test

- Request each Object–specific Services, 75 - 99, addressed to the EM connection Instance.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the EM connection Instance.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

1.5) Send a Delete request to the Explicit Messaging connection.

Pass: The expected response from table below for Delete Request.

If Delete is Implemented	If Delete is NOT Implemented
[Success_Response]	[Service_Not_Supported]

1.6) PreConsumption Timer/Timeout Test

- Open/Allocate an Explicit Messaging (EM) connection.
- Set a test timer for 13 seconds and wait for the test timer to expire.
- Send a Get_Attribute_Single request to the State attribute.
  Pass: state = Non–existent, Error_Response 94 16 FF, Object_Does_Not_Exist
- Open/Allocate an Explicit Messaging (EM) connection.
- Request a Set_Attribute_Single, Expected_Packet_Rate, value = 1 second.
- Set a test timer for (EPR * 4) + 1 seconds and wait for the test timer to expire.
- Send a Get_Attribute_Single request to the State attribute.
  Pass: state = Non–existent, Error_Response 94 16 FF, Object_Does_Not_Exist

1.7) Deferred Delete Test, if the WTA attribute can be set for UCMM connection.

- Open an Explicit Message connection.
- Set Explicit Message EPR = 500 Milliseconds.
- Allocate the I/O connection.
- Set the WTA = Deferred Delete.
- Set the I/O EPR = 1000 Milliseconds.
- Set an internal timer for 3000 milliseconds and wait for the timer to expire.
- Send a Get_Attribute_Single request for the Explicit Message connection State.
  Pass: State = Established.
- Set an internal timer for 2000 milliseconds and wait for the timer to expire.
- Send a Get_Attribute_Single request for the Explicit Message connection State.
  Pass: State = Non-Existent.
- Release the I/O connection.

Part 2: I/O Connection

Note: Each implemented I/O connection shall be tested as described in Part 2: I/O Connection.

State: **Non-Existent**

- Open an Explicit Messaging connection to continue testing for the I/O Connection.

2.1) Send a Delete request to a non-existent connection instance.

Pass: Error_Response: 94 16 FF, Object_Does_Not_Exist or 94 08 FF, Service_Not_Supported

2.2) Attribute Access for Non-Existent instance.

- Send a Get_Attribute_Single request to attribute 1 of the non–existent I/O connection.
  Pass: Error_Response: 94 16 FF, Object_Does_Not_Exist
- Send a Set_Attribute_Single request to attribute 1 of the non–existent I/O connection.
  Pass: Error_Response: 94 16 FF, Object_Does_Not_Exist

2.3) Send a Reset request to the non–existent I/O connection.

Pass: Error_Response: 94 16 FF, Object_Does_Not_Exist or 94 08 FF, Service_Not_Supported

2.4) Send an Apply_Attributes request to the non–existent I/O connection.

Pass: Error_Response: 94 16 FF, Object_Does_Not_Exist or 94 08 FF, Service_Not_Supported

- Create/Allocate the I/O connection.

State: **Configuring**

- Send a Get_Attribute_Single to the I/O connection State.
  Pass: state = Configuring

2.5)

- Send a Delete request to the I/O connection.
  Pass: The expected response from table below for Delete request.

If Delete is Implemented	If Delete is not Implemented
[Success_Response]	[Service_Not_Supported]

- Recreate/reallocate the I/O connection in the configuring state to continue testing.

2.6)

- Request Get_Attribute_Single for the I/O connection instance to access attributes 00 - 99.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
State (01)	[USINT (1)]
Instance_Type (02)	[USINT (1)]
TransportClass_Trigger (03)	[USINT (0..255)]
Produced_Connection_Id (04)	[UINT (x0000..x07F0, xFFFF)] Must be valid ID for Predefined Connection
Consumed_Connection_Id (05)	[UINT (x0000..x07F0, xFFFF)] Must be valid ID for Predefined Connection
Initial_Comm_Characteristics (06)	[USINT Two nibbles each = 0..3 or xF]
Produced_Connection_Size (07)	[UINT]
Consumed_Connection_Size (08)	[UINT]
Expected_Packet_Rate (09)	[UINT (0)]

Attribute Name (ID)	[Expected Values, Responses]
Reserved (10, 11)	[Attribute_Not_Supported]
Watchdog_Timeout_Action (12)	[USINT (0)]
Produced_Connection_Path_Length (13)	[UINT (0)]
Produced_Connection_Path (14)	[EPATH]
Consumed_Connection_Path_Length (15)	[UINT (0)]
Consumed_Connection_Path (16)	[EPATH]
Production_Inhibit_Time (17)	[UINT (0), for COS/Cyclic or Dynamic I/O Connections, Attribute_Not_Supported]
Connection_Timeout_Multiplier (18)	[USINT (0), Attribute_Not_Supported]
Connection_Binding_List (19)	[UINT(size),UINT[size], Attribute_Not_Supported]
Undefined (20..99)	[Attribute_Not_Supported]

- Request Set_Attribute_Single for the I/O connection instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
State (01)	[Attribute_Not_Settable (x0E)]
Instance_Type (02)	[Attribute_Not_Settable]
TransportClass_Trigger (03)	[Success_Response, Attribute_Not_Settable]
Produced_Connection_Id (04)	[Success_Response, Attribute_Not_Settable]
Consumed_Connection_Id (05)	[Success_Response, Attribute_Not_Settable]
Initial_Comm_Characteristics (06)	[Success_Response, Attribute_Not_Settable]
Produced_Connection_Size (07)	[Success_Response, Attribute_Not_Settable]
Consumed_Connection_Size (08)	[Success_Response, Attribute_Not_Settable]
Expected_Packet_Rate (09)	[Success_Response with EPR value (UINT), Attribute_Not_Settable]
Reserved (10, 11)	[Attribute_Not_Supported]
Watchdog_Timeout_Action (12)	[Success_Response, Attribute_Not_Settable]
Produced_Connection_Path_Length (13)	[Attribute_Not_Settable]
Produced_Connection_Path (14)	[Success_Response, Attribute_Not_Settable]
Consumed_Connection_Path_Length (15)	[Attribute_Not_Settable]
Consumed_Connection_Path (16)	[Success_Response, Attribute_Not_Settable]
Production_Inhibit_time (17)	[Success_Response with PIT value (UINT), Attribute_Not_Supported]
Connection_Timeout_Multiplier (18)	[Success_Response, Attribute_Not_Supported]
Connection_Binding_List (19)	[UINT(size),UINT[size], Attribute_Not_Supported]
Undefined (20..99)	[Attribute_Not_Supported]

2.6.1) Production Inhibit Timer (PIT) Rounding, if the PIT is settable

- Send A Set_Attribute_Single to the I/O connection PIT, value = 10..20 milliseconds.
  Pass: Success_Response with value greater than or equal to 10..20.
- Send A Set_Attribute_Single to the I/O connection PIT, value = 125..135 milliseconds.
  Pass: Success_Response with value greater than or equal to 125..135.

- Send A Set_Attribute_Single to the I/O connection PIT, value = 250..260 milliseconds.
  Pass: Success_Response with value greater than or equal to 250..260.
- Send A Set_Attribute_Single to the I/O connection PIT, value = 2045..2055 milliseconds.
  Pass: Success_Response with value greater than or equal to 2045..2055.
  Send A Set_Attribute_Single to the I/O connection PIT, value = 4090..5000 milliseconds.
  Pass: Success_Response with value greater than or equal to 4090..5000.
- Send A Set_Attribute_Single to the I/O connection PIT, value = 32760..32770 milliseconds.
  Pass: Success_Response with value greater than or equal to 32760..32770.
- Send A Set_Attribute_Single to the I/O connection PIT, value = 65525..65535 milliseconds.
  Pass: Success_Response with value greater than or equal to 65525..65535.

2.7) Error tests
Note: These are only done when the attributes can be set.

- Request Set_Attribute_Single, Transport_Class_Trigger,
  values x01, x11, x21, x30, x0F x7F
  Pass: For all values, Error_Response: 94 09 FF, Invalid_Attribute_Value
- Request Set_Attribute_Single, Transport_Class_Trigger, value xF0.
  Pass: Success_Response, device must ignore the production trigger bits.
- Request Set_Attribute_Single, Produced_Connection_ID, values xFF0F, xFFFF.
- Request Set_Attribute_Single, Consumed_Connection_ID, values xFF0F, xFFFF.
- Request Set_Attribute_Single, Initial_Comm_Characteristics, values x04. x40, x25, xFF.
- Request Set_Attribute_Single, Watch_Dog_Timeout_Action, value = xFF.
- Request Set_Attribute_Single, Produced_Connection_Path, value = Class ID 0.
- Request Set_Attribute_Single, Consumed_Connection_Path, value = Class ID 0.
  Pass: For all values, Error_Response: 94 09 FF, Invalid_Attribute_Value.

Note: If Success_Response is returned for any of these Set_Attribute_Single requests, restore the default values returned by the Get_Attribute_Single request before continuing the test.

2.8) Send a Reset request to the I/O connection.
Pass: The expected response from table below for Reset request.

If Reset is Implemented	If Reset is NOT Implemented
[Object_State_Conflict (x0C)]	[Service_Not_Supported (x08)]

2.9) Apply Configuring I/O Connection

- If the connection is a Dynamic connection, configure the attributes except for the IDs.
- Send an Apply_Attributes request.
  Pass: The expected response from table below for Apply_Attributes request.

If Apply_Attributes is Implemented	If Apply_Attributes is NOT Implemented
[Success_Response] and Connection IDs data	[Service_Not_Supported (x08)]

- Send a Get_Attribute_Single request to the State attribute.
  Pass: For Dynamic Connection, state = Waiting_for_Connection_ID, else state = Established

Important: If state = Established, go to step 2.15. Otherwise, continue with step 2.10.

State: **Waiting_for_Connection_ID**

2.10)

- Send a Delete request to the I/O connection.
  Pass: Success_Response
- Send a Get_Attribute_Single request to the State attribute.
  Pass: Error_Response: 94 16 FF, Object_Does_Not_Exist
- Recreate, configure and Apply the I/O connection to continue the test.
- Send a Get_Attribute_Single request to the State attribute.
  Pass: state = Waiting_for_Connection_ID

2.11)

Request Get_Attribute_Single for the I/O connection instance to access attributes 00 - 99.
Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
State (01)	[USINT (2)]
Instance_Type (02)	[USINT (1)]
TransportClass_Trigger (03)	[USINT (0..255)]
Produced_Connection_Id (04)	[UINT (xFFFF)]
Consumed_Connection_Id (05)	[UINT (xFFFF)]
Initial_Comm_Characteristics (06)	[USINT Two nibbles each = 0..3 or xF]
Produced_Connection_Size (07)	[UINT]
Consumed_Connection_Size (08)	[UINT]
Expected_Packet_Rate (09)	[UINT]
Reserved (10, 11)	[Attribute_Not_Supported]
Watchdog_Timeout_Action (12)	[USINT 0..2]
Produced_Connection_Path_Length (13)	[UINT]
Produced_Connection_Path (14)	[EPATH]
Consumed_Connection_Path_Length (15)	[UINT]
Consumed_Connection_Path (16)	[EPATH]
Production_Inhibit_Time (17)	[UINT (0)]
Connection_Timeout_Multiplier (18)	[USINT (0), Attribute_Not_Supported]
Connection_Binding_List (19)	[UINT(size),UINT[size], Attribute_Not_Supported]
Undefined (20..99)	[Attribute_Not_Supported]

- Request Set_Attribute_Single for the I/O connection instance to access attributes 00-99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Get_Attribute_Single request.
  Note: Object_State_Conflict (x0C) is expected when the attribute is not settable in the current state and is settable in another state.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
State (01)	[Attribute_Not_Settable (x0E)]

Attribute Name (ID)	[Expected Responses]
Instance_Type (02)	[Attribute_Not_Settable]
TransportClass_Trigger (03)	[Object_State_Conflict (x0C), Attribute_Not_Settable]
Produced_Connection_Id (04)	[Success_Response]
Consumed_Connection_Id (05)	[Success_Response]
Initial_Comm_Characteristics (06)	[Object_State_Conflict, Attribute_Not_Settable]
Produced_Connection_Size (07)	[Object_State_Conflict, Attribute_Not_Settable]
Consumed_Connection_Size (08)	[Object_State_Conflict, Attribute_Not_Settable]
Expected_Packet_Rate (09)	[Object_State_Conflict, Attribute_Not_Settable]
Reserved (10, 11)	[Attribute_Not_Supported]
Watchdog_Timeout_Action (12)	[Object_State_Conflict, Attribute_Not_Settable]
Produced_Connection_Path_Length (13)	[Attribute_Not_Settable]
Produced_Connection_Path (14)	[Object_State_Conflict, Attribute_Not_Settable]
Consumed_Connection_Path_Length (15)	[Attribute_Not_Settable]
Consumed_Connection_Path (16)	[Object_State_Conflict, Attribute_Not_Settable]
Production_Inhibit_Time (17)	[Object_State_Conflict, Attribute_Not_Settable]
Connection_Timeout_Multiplier (18)	[Object_State_Conflict, Attribute_Not_Supported]
Connection_Binding_List (19)	[UINT(size),UINT[size], Attribute_Not_Supported]
Undefined (20..99)	[Attribute_Not_Supported]

2.12) Error tests

Note: These are only done when the attributes can be set.

- Send a Set_Attribute_Single request, Produced_Connection_ID, value = xFF0F.
- Send a Set_Attribute_Single request, Consumed_Connection_ID, value = xFF0F.
  Pass: For all values, Error_Response: 94 09 FF, Invalid_Attribute_Value

Note: If Success_Response is returned for any of these Set_Attribute_Single requests,

Send and Apply_Attributes request

Pass: Error_Response: 94 09 XX, where XX = the attribute ID of the offending attribute.

2.13) Send a Reset request to the I/O connection.

Pass: Error_Response 94 0C FF Object_State_Conflict**,** or 94 08 FF, Service_Not_Supported

2.14) Set Ids, Apply, Waiting Connection Id

- Configure the connection IDs
- Send an Apply_Attributes request to the I/O connection.
  Pass: Success_Response and the ID data
- Send a Get_Attribute_Single request to the State attribute.
  Pass: state = Established

State: **Established**

2.15)

- Send a Delete request to the I/O connection.
  Pass: The expected response from table below for Delete request.

If Delete is Implemented	If Delete is not Implemented
[Success_Response]	[Service_Not_Supported]

- Send a Get_Attribute_Single request to the State attribute of the I/O connection.
  Pass: Error_Response: 94 16 FF, Object_Does_Not_Exist
  Recreate/reallocate the I/O connection and transition it to the Established state.
- Send a Get_Attribute_Single request the State.
  Pass: state = Established

2.16)

- Request Get_Attribute_Single for the I/O connection instance to access attributes 00 - 99.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
State (01)	[USINT (3)]
Instance_Type (02)	[USINT (1)]
TransportClass_Trigger (03)	[USINT (0..255)]
Produced_Connection_Id (04)	[UINT (x0000..x07F0, xFFFF)]
Consumed_Connection_Id (05)	[UINT (x0000..x07F0, xFFFF)]
Initial_Comm_Characteristics (06)	[USINT Two nibbles each = 0..3 or xF]
Produced_Connection_Size (07)	[UINT]
Consumed_Connection_Size (08)	[UINT]
Expected_Packet_Rate (09)	[UINT]
Reserved (10, 11)	[Attribute_Not_Supported]
Watchdog_Timeout_Action (12)	[USINT 0..2]
Produced_Connection_Path_Length (13)	[UINT]
Produced_Connection_Path (14)	[EPATH]
Consumed_Connection_Path_Length (15)	[UINT]
Consumed_Connection_Path (16)	[EPATH]
Production_Inhibit_Time (17)	[UINT]
Connection_Timeout_Multiplier (18)	[UINT(0), Attribute_Not_Supported]
Connection_Binding_List (19)	[UINT(size),UINT[size], Attribute_Not_Supported]
Undefined (20..99)	[Attribute_Not_Supported]

- Request Set_Attribute_Single for the I/O connection instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Get_Attribute_Single request.
  Note: Object_State_Conflict (x0C) is expected when the attribute is not settable in the current state and is settable in another state.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
State (01)	[Attribute_Not_Settable (x0E)]
Instance_Type (02)	[Attribute_Not_Settable]
TransportClass_Trigger (03)	[Object_State_Conflict (x0C), Attribute_Not_Settable]

Attribute Name (ID)	[Expected Responses]
Produced_Connection_Id (04)	[Success_Response, Attribute_Not_Settable]
Consumed_Connection_Id (05)	[Success_Response, Attribute_Not_Settable]
Initial_Comm_Characteristics (06)	[Object_State_Conflict, Attribute_Not_Settable]
Produced_Connection_Size (07)	[Object_State_Conflict, Attribute_Not_Settable]
Consumed_Connection_Size (08)	[Object_State_Conflict, Attribute_Not_Settable]
Expected_Packet_Rate (09)	[Success_Response with EPR value (UINT)]
Reserved (10, 11)	[Attribute_Not_Supported]
Watchdog_Timeout_Action (12)	[Success_Response, Attribute_Not_Settable]
Produced_Connection_Path_Length (13)	[Attribute_Not_Settable]
Produced_Connection_Path (14)	[Object_State_Conflict, Attribute_Not_Settable]
Consumed_Connection_Path_Length (15)	[Attribute_Not_Settable]
Consumed_Connection_Path (16)	[Object_State_Conflict, Attribute_Not_Settable]
Production_Inhibit_Time (17)	[Success_Response with PIT value (UINT) , Attribute_Not_Settable]
Connection_Timeout_Multiplier (18)	[Success_Response, Attribute_Not_Supported]
Connection_Binding_List (19)	[UINT(size),UINT[size], Attribute_Not_Supported]
Undefined (20..99)	[Attribute_Not_Supported]

2.16.1) Verify Connection Paths

- Send a Get_Attribute_Single request for the Produced and Consumed Connection Path.
- Verify that the path contains valid data when produced/consumed size is non-zero.
  Pass: Path length is correct and, for Logical Path, Class and Instance are valid

2.17) Send an Apply_Attributes request to the I/O connection.

Pass: Error_Response 94 0C FF Object_State_Conflict**,** or 94 08 FF, Service_Not_Supported

2.18) Reset Established I/O connection, If Reset is not supported. Go to Step 20.

- Set the I/O connection, Watchdog_Timeout_Action = Transition_To_Timed_Out
- Send a Get_Attribute_Single request to the I/O connection, Watchdog_Timeout_Action.
- Set the I/O connection EPR, = 1.25 seconds.
- Send I/O data of size specified by the Consumed Connection Size attribute.
- Set a test timer for (EPR * 4) - .2 seconds and wait for the test timer to expire.
- Send a Get_Attribute_Single request to the State attribute.
  Pass: state = Established
- Send a Reset request to the I/O connection.
- Set a test timer to expire after (EPR * 4) - .2 seconds and wait for the test timer to expire.
- Send a Get_Attribute_Single request to the State attribute.
  Pass: state = Established

2.19) Reset Timed Out I/O connection

- Set a test timer for 2 seconds and wait for the test timer to expire.
- Send a Get_Attribute_Single request to the State attribute.
  Pass: state = Timed_Out
- Send a Reset request to the I/O connection.
- Send a Get_Attribute_Single request to the State attribute.
  Pass: state = Established

2.20) Error tests (Only done if the attributes can be set.)

- Send a Set_Attribute_Single request, Produced_Connection_ID, value = xFF0F.
- Send a Set_Attribute_Single request, Consumed_Connection_ID, value = xFF0F.
- Request Set_Attribute_Single, Watch_Dog_Timeout_Action, value = xFF.
  Pass: For all values, Error_Response: 94 09 FF, Invalid_Attribute_Value

2.21) Transition to TimedOut

- Set the I/O connection EPR = 250 milliseconds.
- Set a test timer for (Actual EPR + 300ms) seconds and wait for the test timer to expire.
- Send a Get_Attribute_Single request to the State attribute.
  Pass: state = Watchdog_Timeout_Action state

State: **Timed_Out**

2.22) **Note:** If the state is not Timed Out, go to Part 3.

- Request Get_Attribute_Single for the I/O connection instance to access attributes 00 - 99.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
State (01)	[USINT (4)]
Instance_Type (02)	[USINT (1)]
TransportClass_Trigger (03)	[USINT (0..255)]
Produced_Connection_Id (04)	[UINT (x0000..x07F0, xFFFF)]
Consumed_Connection_Id (05)	[UINT (x0000..x07F0, xFFFF)]
Initial_Comm_Characteristics (06)	[USINT Two nibbles each = 0..3 or xF]
Produced_Connection_Size (07)	[UINT]
Consumed_Connection_Size (08)	[UINT]
Expected_Packet_Rate (09)	[UINT]
Reserved (10, 11)	[Attribute_Not_Supported (x14)]
Watchdog_Timeout_Action (12)	[USINT 0..2]
Produced_Connection_Path_Length (13)	[UINT]
Produced_Connection_Path (14)	[EPATH]
Consumed_Connection_Path_Length (15)	[UINT]
Consumed_Connection_Path (16)	[EPATH]
Production_Inhibit_Time (17)	[Success_Response / PIT value (UINT) , Attribute_Not_Settable]
Connection_Timeout_Multiplier (18)	[USINT (0), Attribute_Not_Supported]
Connection_Binding_List (19)	[UINT(size),UINT[size], Attribute_Not_Supported]
Undefined (20..99)	[Attribute_Not_Supported]

- Request Set_Attribute_Single for the I/O connection instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.
  Note: Object_State_Conflict (x0C) is expected when the attribute is not settable in the current state and is settable in another state.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
State (01)	[Attribute_Not_Settable]
Instance_Type (02)	[Attribute_Not_Settable]
TransportClass_Trigger (03)	[Object_State_Conflict (x0C), Attribute_Not_Settable]
Produced_Connection_Id (04)	[Success_Response, Attribute_Not_Settable]
Consumed_Connection_Id (05)	[Success_Response, Attribute_Not_Settable]
Initial_Comm_Characteristics (06)	[Object_State_Conflict, Attribute_Not_Settable]
Produced_Connection_Size (07)	[Object_State_Conflict, Attribute_Not_Settable]
Consumed_Connection_Size (08)	[Object_State_Conflict, Attribute_Not_Settable]
Expected_Packet_Rate (09)	[Service_Success with EPR value (UINT)]
Reserved (10, 11)	[Attribute_Not_Supported]
Watchdog_Timeout_Action (12)	[Success_Response, Attribute_Not_Settable]
Produced_Connection_Path_Length (13)	[Attribute_Not_Settable]
Produced_Connection_Path (14)	[Object_State_Conflict, Attribute_Not_Settable]
Consumed_Connection_Path_Length (15)	[Attribute_Not_Settable]
Consumed_Connection_Path (16)	[Object_State_Conflict, Attribute_Not_Settable]
Production_Inhibit_Time (17)	[Success_Response with PIT value (UINT), Attribute_Not_Settable]
Connection_Timeout_Multiplier (18)	[Success_Response, Attribute_Not_Supported]
Connection_Binding_List (19)	[UINT(size),UINT[size], Attribute_Not_Supported]
Undefined (20..99)	[Attribute_Not_Supported]

2.23) Send an Apply_Attributes request to the I/O connection.

Pass: Error_Response 94 0C FF Object_State_Conflict**,** or 94 08 FF, Service_Not_Supported

2.24) Error tests

Note: These are only done when the attributes can be set.

- Send a Set_Attribute_Single request, Produced_Connection_ID, value = xFF0F.
- Send a Set_Attribute_Single request, Consumed_Connection_ID, value = xFF0F.
- Request Set_Attribute_Single, Watch_Dog_Timeout_Action, value = xFF.
  Pass: For all values, Error_Response: 94 09 FF, Invalid_Attribute_Value

2.25)

- Send a Delete request to the I/O connection.
  Pass: The expected response from table below for Delete request.

If Delete is Implemented	If Delete is not Implemented
[Success_Response]	[Service_Not_Supported]

2.26) If the WatchDog_Timeout_Action is not settable skip this step.

2.26.1) WatchDog_Timeout_Action, Auto_Reset.

- Request a Set_Attribute_Single, WatchDog_Timeout_Action, value = Auto_Reset
- If the response is Success_Response, complete this step. Otherwise, go to step 2.26.2.
- Reset/Allocate an I/O connection, if necessary, Create and configure the connection.
- Request a Set_Attribute_Single, Expected_Packet_Rate, value = 250 milliseconds.

- Set a test timer for (Actual EPR * 4) + 100 milliseconds and wait for the test timer to expire.
- Send a Get_Attribute_Single request to the State attribute.
  Pass: state = Established

2.26.2) WatchDog_Timeout_Action, Auto_Delete

- Request a Set_Attribute_Single, WatchDog_Timeout_Action, value = Auto_Delete
- If the response is Success_Response, complete this test. Otherwise, go to Part 3.
- Reset/Allocate an I/O connection, if necessary, Create and configure the connection.
- Request a Set_Attribute_Single, Expected_Packet_Rate, value = .25 second.
- Set a test timer for (EPR * 4) + 100 milliseconds and wait for the test timer to expire.
- Send a Get_Attribute_Single request to the State attribute.
  Pass: state = Non–existent, or Error_Response 94 16 FF, Object_Does_Not_Exist
  Pass: state = Established, for Producing only client connections.

2.27) Check Production Inhibit <= EPR at Apply

- Create/Allocate an I/O connection, if necessary, configure the connection.
- Request a Set_Attribute_Single, Production_Inhibit_Timer, value = 2 seconds.
- Request a Set_Attribute_Single, Expected_Packet_Rate, value = 1 second.
- If necessary, request Apply_Attributes.
  Pass: Error_Response 94 09 11, Invalid_Attribute_Value, Production_Inhibit_Timer

Part 3: Class level testing

Note: The Explicit Message connection is in the Established state and is used for class level testing.

1) EPR Behavior test.

- Open an Explicit Message Connection
- Set the EPR = 10 milliseconds.
- Set a test timer for 30 milliseconds. Wait for timer to expire.
- Send a Get_Attribute_Single to the EM connection State.
  Pass: state = Established
- Send A Set_Attribute_Single to the EM connection EPR, value = 1 millisecond.
  Pass: Success_Response with EPR value greater than or equal to 1.
- Set a test timer for (Actual EPR * 4) + 2 seconds. Wait for timer to expire.
- Send a Get_Attribute_Single request to the EM connection State.
  Pass: state = Non_Existent, or Error_Response 94 16 FF, Object_Does_Not_Exist
- Open an Explicit Message Connection
- Open an I/O connection
- Set the I/O connection EPR = 10 milliseconds.
- Send I/O data of size specified by the Consumed Connection Size attribute.
- Set a test timer for 30 milliseconds. Wait for timer to expire.
- Send a Get_Attribute_Single to the I/O connection State.
  Pass: state = Established
- Set the I/O connection EPR value = 1 millisecond
  Pass: Success_Response with EPR value greater than or equal to 1.

3.2) EPR timer rounding.

- Send A Set_Attribute_Single to the EM connection EPR, value = 10..20 milliseconds.

Pass: Success_Response with EPR value greater than or equal to 10..20.

- Send A Set_Attribute_Single to the EM connection EPR, value = 125..135 milliseconds.
  Pass: Success_Response with EPR value greater than or equal to 125..135.
- Send A Set_Attribute_Single to the EM connection EPR, value = 250..260 milliseconds.
  Pass: Success_Response with EPR value greater than or equal to 250..260.
- Send A Set_Attribute_Single to the EM connection EPR, value = 2045..2055 milliseconds.
  Pass: Success_Response with EPR value greater than or equal to 2045..2055.

- Send A Set_Attribute_Single to the EM connection EPR, value = 4090..5000 milliseconds.
  Pass: Success_Response with EPR value greater than or equal to 4090..5000.
- Send A Set_Attribute_Single to the EM connection EPR, value = 32760..32770 milliseconds.
  Pass: Success_Response with EPR value greater than or equal to 32760..32770.
- Send A Set_Attribute_Single to the EM connection EPR, value = 65525..65535 milliseconds.
  Pass: Success_Response with EPR value greater than or equal to 65525..65535.
  Note: If overflow occurs, the maximum value is accepted. Stop EPR timer rounding test.

3.3) Maximum EPR Test

- Set the EPR = 65535. (use 65535 or max value accepted by device).
- Set a test timer for (Actual EPR * 4) - 2 seconds.
- When timer expires, send a Get_Attribute_Single to the EM connection, State attribute.
  Pass: state = Established
- Open an Explicit Messaging Connection.
- Send a Set_Attribute_Single request to the Explicit Message connection EPR, value = 0.

3.4) Class Reset Test. Omit this step if Reset is not supported at the class level.

- Open one or more I/O Connections.
- Send a Set_Attribute_Single to the I/O connection EPR, value = 1.25 seconds.
- Send I/O data of size specified by the Consumed Connection Size attribute.
- Set a test timer for (Actual EPR * 4) + 2 seconds and wait for the test timer to expire.
- Send a Reset request to the Connection Class, instance zero.
  Pass: Success_Response
- Send a Get_Attribute_Single to the EM and I/O connection(s), State attribute.
  Pass: state = Established, for Explicit Message and each I/O connection.
- Set a test timer for 4 seconds and wait for the test timer to expire.
- Send a Reset request to the Connection Class, instance zero.
  Pass: Success_Response
- Set a test timer for 4 seconds and wait for the test timer to expire.
- Send a Get_Attribute_Single to the I/O connection(s), State attribute.
  Pass: state = Established, for each I/O connection

Close the I/O connections.

3.5) Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00-99. Save the returned value for each Get_Attribute_Single request.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.

Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT (1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT (16..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT (16..199), Service_Not_Supported, Attribute_Not_Supported]
Connection_Request_Error_Count (08)	[Service_Not_Supported, Attribute_Not_Supported] Note 1
Safety_Connection_Counters (09)	[Service_Not_Supported, Attribute_Not_Supported] Note 1
Undefined (10..99)	[Service_Not_Supported, Attribute_Not_Supported]

Note 1: For Safety Devices, See CIP Safety PCTS (ODVA PUB00170) for test procedures

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00-99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

3.6) Common Services Test

- Request each of the Common Services 00-49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
(0..4)	[Service_Not_Supported (x08)]
Reset (05)	[Success_Response, Service_Not_Supported]
(6, 7)	[Service_Not_Supported (x08)]
Create (08)	[Success_Response, Service_Not_Supported]
Delete (09)	[Success_Response, Service_Not_Supported]
(10..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response, Attribute_Not_Supported (x14), Service_Not_Supported]
(15, 16)	[Service_Not_Supported]
Find_Next_Object_Instance (17)	[Success_Response, Service_Not_Supported]
(18..49)	[Service_Not_Supported]

3.7) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75-99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Connection_Bind (75)	[Service_Not_Supported (x08)]

Producing_Application_Lookup (76)	[Service_Not_Supported (x08)]
Reset_Counters (77)	[Service_Not_Supported (x08)] Note 1
Safety Close (78)	[Service_Not_Supported (x08)] Note 1
Object–specific Codes (79..83)	[Service_Not_Supported (x08)]
Safety Open (84)	[Service_Not_Supported (x08)] Note 1
Object–specific Codes (85..99)	[Service_Not_Supported (x08)]

Note 1: For Safety Devices, See CIP PCTS (ODVA PUB00170) for test procedures

- Request each Reserved Service, 100-255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

3.8) Class Delete Test, If Delete is implemented for the class,

- Open an I/O connection.
- Request Delete for the Class
- Send a Get_Attribute_Single to the Explicit Message and I/O connection(s), State attribute.
  Pass: No_Response

3.9) If Find_Next_Object_Instance is implemented for the class,

- request a Find_Next_Object_Instance, Instance ID = 0, Max_Return_Values = 255.
  Pass: Success_Response, first USINT = size, Array of UINT[size].
- request a Find_Next_Object_Instance, Instance ID = Array[size], Max_Return_Values 255.
  Pass: Success_Response, USINT = 0.
- request a Find_Next_Object_Instance, Instance ID = 0, Max_Return_Values = 1.
  Pass: Success_Response, first USINT = 1, Array[1] greater than or = 1.
  Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.5 Connection Manager Object Test x06

This section defines the conformance test for the Connection Manager object. This test is required when the Connection Manager object is implemented in the device.

Functional Description

This test verifies the:

Part 1) Class attributes access rules for the Connection Manager object.

Part 2) Instance attribute access rules for the Connection Manager object.

Part 3) Common Service implementations for class and instance.

Part 4) Object–specific and Reserved service implementations for class and instance.

Part 5) Conformance when incorrect data is used to access attributes and services.

Part 6) Object behavior accessible from the network.

6.1 Forward Close

6.2 Unconnected Send

6.3 Forward Open

6.4 GetConnectionData

6.5 GetConnectionOwner

6.6 Class 3 Connected Explicit Messaging

Part 7) I/O test

Procedure Definition

- Initialization
  The Connection Manager object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT , Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attributes (06)	[UINT (1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attributes (07)	[UINT (1..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Open Requests (01)	[UINT16, Attribute_Not_Supported, Service_Not_Supported]
Open Format Rejects (02)	[UINT16, Attribute_Not_Supported, Service_Not_Supported]
Open Resource Rejects (03)	[UINT16, Attribute_Not_Supported, Service_Not_Supported]
Open Other Rejects (04)	[UINT16, Attribute_Not_Supported, Service_Not_Supported]
Close Requests (05)	[UINT16, Attribute_Not_Supported, Service_Not_Supported]
Close Format Rejects (06)	[UINT16, Attribute_Not_Supported, Service_Not_Supported]
Close Other Rejects (07)	[UINT16, Attribute_Not_Supported, Service_Not_Supported]
Connection Timeouts (08)	[UINT16, Attribute_Not_Supported, Service_Not_Supported]
Connection Entry List (09)	[Struct, Attribute_Not_Supported, Service_Not_Supported]
CPU_Utilization (11)	[UINT16, Attribute_Not_Supported, Service_Not_Supported]
MaxBuffSize (12)	[UDINT, Attribute_Not_Supported, Service_Not_Supported]
BufSize Remaining(13)	[UDINT, Attribute_Not_Supported, Service_Not_Supported]
Max Connection Establishment Time (14)	[UINT16, Attribute_Not_Supported, Service_Not_Supported]
I/O Packets per Second (15)	[UDINT, Attribute_Not_Supported, Service_Not_Supported]
Percent I/O Utilization (16)	[UINT16, Attribute_Not_Supported, Service_Not_Supported]
Explicit Packets per Second (17)	[UDINT, Attribute_Not_Supported, Service_Not_Supported]
Missed I/O Packets (18)	[UDINT, Attribute_Not_Supported, Service_Not_Supported]
CIP I/O Connections (19)	[UDINT, Attribute_Not_Supported, Service_Not_Supported]
CIP Explicit Connections (20)	[UDINT, Attribute_Not_Supported, Service_Not_Supported]
Undefined (21..99)	[Attribute_Not_Supported, Service_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Open Requests (01)	[Service_Not_Supported, Invalid_Attribute_Value (x09), Success]
Open Format Rejects (02)	[Service_Not_Supported, Invalid_Attribute_Value (x09), Success]
Open Resource Rejects (03)	[Service_Not_Supported, Invalid_Attribute_Value (x09), Success]
Open Other Rejects (04)	[Service_Not_Supported, Invalid_Attribute_Value (x09), Success]

Close Requests (05)	[Service_Not_Supported, Invalid_Attribute_Value (x09), Success]
Close Format Rejects (06)	[Service_Not_Supported, Invalid_Attribute_Value (x09), Success]
Close Other Rejects (07)	[Service_Not_Supported, Invalid_Attribute_Value (x09), Success]
Connection Timeouts (08)	[Service_Not_Supported, Invalid_Attribute_Value (x09), Success]
Connection Entry List (09)	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
CPU_Utilization (11)	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
MaxBuffSize (12)	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
BufSize Remaining(13)	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
Max Connection Establishment Time (14)	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
I/O Packets per Second (15)	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
Percent I/O Utilization (16)	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
Explicit Packets per Second (17)	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
Missed I/O Packets (18)	[Service_Not_Supported, Invalid_Attribute_Value (x09), Success]
CIP I/O Connections (19)	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
CIP Explicit Connections (20)	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
Undefined (21..99)	[Service_Not_Supported, Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attribute_All (01)	[all attribute values, Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] if no attributes are supported
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attribute_All (01)	[all attribute values, Service_Not_Supported]
(02..13)	[Service_Not_Supported (x08)]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]

Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.
None

6) Behavior tests.

6.1) Forward Close

Run this test to both Class 1 (if supported) and Class 3 connection.

For Class 1 connection, it sends ForwardOpen with optional SockAddr Info items in both O->T and T->O direction and the sin_addr is set to an invalid IP address.

Pass: The device can handle these optional items correctly and open the connection successfully.

Fail: The DUT refuses to open connection due to these optional SockAddrInfo items.

a) Normal Forward Close
Issues ForwardOpen, then ForwardClose
Pass: Successful Forward Close

b) Non-Existant Connection
ForwardClose when no ForwardOpen hass been issued.
Pass: Unsuccessful ForwardClose with extended status 0x107

c) Path wrong size
This test case is now covered by Message Router test MR.2.E Request Path Size too small.

d) Wrong Class Segment
This test case is now covered by Message Router test MR.2.E Wrong Class Segment.

e) Wrong Class Value
This test case is now covered by Message Router test MR.2.E Wrong Class Number.

f) Wrong Logical Segment
This test case is now covered by Message Router test MR.2.E Wrong Logical Segment.

g) Wrong Logical Value
This test case is now covered by Message Router test MR.2.E Wrong Instance (logical number).

h) Connection Path Size Too Small
Pass: Successful ForwardClose, 0x15 (Too much data)

i) Connection Path Size Too Large
Pass: Successful ForwardClose, 0x13 (Not enough data)

j) Wrong Class Segment
Pass: Successful ForwardClose, [0x1,0x315 (Invalid Segment in Connection Path)] or [(0x1, 0x316 (Error in Forward Close)]

k) Wrong Conn Class Value
Pass: Successful ForwardClose, [0x1,0x315 (Invalid Segment in Connection Path)] or [(0x1, 0x316 (Error in Forward Close)]

l) Wrong Conn Logical Value
Pass: Successful ForwardClose, [0x1,0x315 (Invalid Segment in Connection Path)] or [(0x1, 0x316 (Error in Forward Close)]

m) Wrong Conn Logical Segment
Pass: Successful ForwardClose, [0x1,0x315 (Invalid Segment in Connection Path)] or [(0x1, 0x316 (Error in Forward Close)]

n) Wrong IP Address

Pass: DUT responds with status 0x0F (PRIVILEGE VIOLATION) to a Forward Close received from an IP Address other than the one on which the Forward Open was received.

NOTE: *this test requires a specific additional address configured on the selected host interface at the NEXT contiguous address; e.g., if the testing interface is at 192.168.1.100, Wrong IP Address Forward Close will be sent from 192.168.1.101. Use the Internet Protocol (IPv4) Properties for the testing network connection to add this IP Address using the Advanced… option for IP Settings.*

6.2) Unconnected Send

This test is performed if the Unconnected_Send service is implemented or CIP routing support is indicated in the STC. An error is logged if service is implemented and CIP routing support is not indicated. If CIP routing support is claimed, an error is logged if the Port object is not implemented or if an instance of the Port object cannot be located for the DUT entry port.

a) Retrieve entry port

Pass: Successful Get_Attribute_Single of Port class attribute Port Entry Port

b) Determine route to self

Pass: Port instance located for the entry port

c) Issue Unconnected Send request using self route to retrieve Vendor ID

Pass: Successful Unconnected Send response

This test is under development. Contact ODVA for details.

6.3) ForwardOpen/LargeForwardOpen

Perform the following tests (Connection Establishment, Electronic Key Validation, Connection Size Validation) using ForwardOpen (Service Code = 54_{hex}) and LargeForwardOpen (Service Code = $5B_{hex}$), if supported by device. Use “normal” and “error” values for O→T and T→O Network Connection Parameters as specified in the following table:

Parameter	FO Normal	FO Error	LargeFO Normal	LargeFO Error
O→T Network Connection Parameters	0x420A	0x620A		
T→O Network Connection Parameters	0x4240	0x6240		

6.3.1) Connection Establishment

a) FwdOpen(ConnPath = MsgRouter, OT/TO = Normal, TranTrigger = 0xA3);
Issue ForwardOpen, issue Get_Attribute_Single on Identity instance 1, close with ForwardClose

b) FwdOpen(Duplicate);
Issue two identical ForwardOpen commands. The second one should fail with general status 1 and extended status 0x0100, then close with ForwardClose.

c) FwdOpen(Reserved Connection Type);
expects: extended status Invalid Network Connection Type (O2T: 0x0123, T2O: 0x0124) – warn for deprecated extended status Invalid Network Connection Parameter (0x0108)

d) FwdOpen(Invalid Production Trigger);
trigger = 0x43
expects: extended status Production Trigger Not Supported (0x011D), Direction Not Supported (0x011E), or Transport Class and Trigger Combination Not Supported (0x0103)

e) FwdOpen(Conn too Small);
Connection Path Size = 1
expects TOO_MUCH_DATA

f) FwdOpen(Conn too Big);
Connection Path Size = 3
expects NOT_ENOUGH_DATA

g) FwdOpen(Wrong Conn Class Segment);
Connection Class Segment = FF
expects general status 1, extended status 0x0315

h) FwdOpen(Wrong Conn Class Value);
Connection Path class value = FF
expects general status 1, extended status 0x0315

i) FwdOpen(Wrong Conn Logical Segment);
Connection Path instance logical segment = FF
expects general status 1, extended status 0x0315

j) FwdOpen(Wrong Conn Logical Instance);
Connection path instance = FF
expects general status 1, extended status 0x0315

k) FwdOpen(Connection Timeout Test);
ConnPath = MsgRouter, OT/TO = Normal, TranTrigger = 0xA3
expects: Alive at 9 Seconds after Get_Attribute_Single on Identity instance 1 succeeds.

6.3.2) Electronic Key Validation

a) FwdOpen(Valid Electronic Key);
Expects: success

b) FwdOpen(Electronic Key Exact Match, Vendor ID = 0);
Expects: success

c) FwdOpen(Electronic Key Exact Match, Product Type = 0);
Expects: success

d) FwdOpen(Electronic Key Exact Match, Product Code = 0);
Expects: Success

e) FwdOpen(Electronic Key Exact Match, Invalid Vendor ID);
Expects: Unsuccessful ForwardOpen, status = 1, extended status = 0x114

f) FwdOpen(Electronic Key Exact Match, Invalid Product Type);
Expects: Unsuccessful Forward Open, status 1, extended status = 0x115

g) FwdOpen(Electronic Key Exact Match, Invalid Product Code);
Expects: Unsuccessful forward open, extended status 0x114

h) FwdOpen(Electronic Key Exact Match, Key Format = 3 (Reserved));
Expects: Unsuccessful forward open, extended status 0x0x315

i) FwdOpen(Electronic Key Exact Match, Key Format = 5 (Reserved));
Expects: Unsuccessful forward open, extended status 0x315

j) FwdOpen(Electronic Key Exact Match, MajRev = 0, MinRev=0);
Expects: Success

k) FwdOpen(Electronic Key Exact Match, MajRev = 0, MinRev=SMALL[1]);
Expects: Success

l) FwdOpen(Electronic Key Exact Match, MajRev = 0, MinRev=Correct);
Expects: Sucess

m) FwdOpen(Electronic Key Exact Match, MajRev = 0, MinRev=BIG);
Expects: Success

n) FwdOpen(Electronic Key Exact Match, MajRev = SMALL[1], MinRev=0);
Expects: Unsuccessful Forward open with extended status 0x116

o) FwdOpen(Electronic Key Exact Match, MajRev = SMALL[1], MinRev = SMALL[1]);
Expects: Unsuccessful Forward open with extended status 0x116

[1] NOTE: for all SMALL Major and Minor revision variation tests: the test shall be performed ONLY IF the corresponding actual value in the Device Under Test is greater than one (1).

p) FwdOpen(Electronic Key Exact Match, MajRev = SMALL[1], MinRev = Correct);
Expects: Unsuccessful Forward open with extended status 0x116

q) FwdOpen(Electronic Key Exact Match, MajRev = SMALL[1], MinRev = BIG);
Expects: Unsuccessful Forward open with extended status 0x116

r) FwdOpen(Electronic Key Exact Match, MajRev = Correct, MinRev=0);
Expects: success

s) FwdOpen(Electronic Key Exact Match, MajRev = Correct, MinRev = SMALL[1]);
Expects: Unsuccessful Forward open with extended status 0x116

t) FwdOpen(Electronic Key Exact Match, MajRev = Correct, MinRev = BIG);
Expects: Unsuccessful Forward open with extended status 0x116

u) FwdOpen(Electronic Key Exact Match, MajRev = BIG, MinRev=0);
Expects: Unsuccessful Forward open with extended status 0x116

v) FwdOpen(Electronic Key Exact Match, MajRev = BIG, MinRev = SMALL[1]);
Expects: Unsuccessful Forward open with extended status 0x116

w) FwdOpen(Electronic Key Exact Match, MajRev = BIG, MinRev = Correct);
Expects: Unsuccessful Forward open with extended status 0x116

x) FwdOpen(Electronic Key Exact Match, MajRev = BIG, MinRev = BIG);
Expects: Unsuccessful Forward open with extended status 0x116

y) FwdOpen(Electronic Key Compatible Match, MajRev = 0, MinRev=0);
Expects: Unsuccessful Forward open with extended status 0x116

z) FwdOpen(Electronic Key Compatible Match, MajRev = 0, MinRev=SMALL[1]);
Expects: Unsuccessful Forward open with extended status 0x116

aa) FwdOpen(Electronic Key Compatible Match, MajRev = 0, MinRev=Correct);
Expects: Unsuccessful Forward open with extended status 0x116

bb) FwdOpen(Electronic Key Compatible Match, MajRev = 0, MinRev=BIG);
Expects: Unsuccessful Forward open with extended status 0x116

cc) FwdOpen(Electronic Key Compatible Match, MajRev = SMALL[1], MinRev=0);
Expects: Unsuccessful ForwardOpen with extended status 0x116

dd) FwdOpen(Electronic Key Compatible Match, MajRev = SMALL[1], MinRev = SMALL[1]);
Expects: Unsuccessful ForwardOpen with extended status 0x116, or Success

ee) FwdOpen(Electronic Key Compatible Match, MajRev = SMALL[1], MinRev = Correct);
Expects: Unsuccessful ForwardOpen with extended status 0x116, or Success

ff) FwdOpen(Electronic Key Compatible Match, MajRev = SMALL[1], MinRev = BIG);
Expects: Unsuccessful ForwardOpen with extended status 0x116, or Success

gg) FwdOpen(Electronic Key Compatible Match, MajRev = Correct, MinRev=0);
Expects: Unsuccessful ForwardOpen with extended status 0x116

hh) FwdOpen(Electronic Key Compatible Match, MajRev = Correct, MinRev = SMALL[1]);
Expects: Success

ii) FwdOpen(Electronic Key Compatible Match, MajRev = Correct, MinRev = BIG);
Expects: Unsuccessful ForwardOpen with extended status 0x116

jj) FwdOpen(Electronic Key Compatible Match, MajRev = BIG, MinRev=0);
Expects: Unsuccessful ForwardOpen with extended status 0x116

kk) FwdOpen(Electronic Key Compatible Match, MajRev = BIG, MinRev = SMALL[1]);
Expects: Unsuccessful ForwardOpen with extended status 0x116

ll) FwdOpen(Electronic Key Compatible Match, MajRev = BIG, MinRev = Correct);
Expects: Unsuccessful ForwardOpen with extended status 0x116

mm) FwdOpen(Electronic Key Compatible Match, MajRev = BIG, MinRev = BIG);
Expects: Unsuccessful ForwardOpen with extended status 0x116

nn) FwdOpen(NULL Electronic Key, MajRev = 0, MinRev = 0);
Expects: Success

6.3.3) Connection Size Validation

For each fixed-size I/O connection, vary the Network Connection Parameter connection size in the ForwardOpen/LargeForwardOpen request to simulate wrongly configured connections due to 16-bit sequence number and 32-bit real time header overhead.

Use the first supported trigger type declared for the connection. Adjust the connection size as described below. Each test variation shall only be performed if the adjusted size is greater than zero. If the fixed connection size is in a range, the test size is calculated based on the given min/max value.

Direction	Size Variation	[Expected Responses][2]
O2T	-2	[Invalid_Connection_Size (x01, 0x0127, warn if 0x0109 or 0x0315 returned)]
O2T	+2	[Invalid_Connection_Size (x01, 0x0127, warn if 0x0109 or 0x0315 returned)]
O2T	-4	[Invalid_Connection_Size (x01, 0x0127, warn if 0x0109 or 0x0315 returned)]
O2T	+4	[Invalid_Connection_Size (x01, 0x0127, warn if 0x0109 or 0x0315 returned)]
O2T	-6	[Invalid_Connection_Size (x01, 0x0127, warn if 0x0109 or 0x0315 returned)]
O2T	+6	[Invalid_Connection_Size (x01, 0x0127, warn if 0x0109 or 0x0315 returned)]
T2O	-2	[Invalid_Connection_Size (x01, 0x0128, warn if 0x0109 or 0x0315 returned)]
T2O	+2	[Invalid_Connection_Size (x01, 0x0128, warn if 0x0109 or 0x0315 returned)]
T2O	-4	[Invalid_Connection_Size (x01, 0x0128, warn if 0x0109 or 0x0315 returned)]
T2O	+4	[Invalid_Connection_Size (x01, 0x0128, warn if 0x0109 or 0x0315 returned)]
T2O	-6	[Invalid_Connection_Size (x01, 0x0128, warn if 0x0109 or 0x0315 returned)]
T2O	+6	[Invalid_Connection_Size (x01, 0x0128, warn if 0x0109 or 0x0315 returned)]

6.3.4) Connection ID Usage

This test attempts to verify the following requirement:

"The Network Connection ID shall not be reused until the connection has been closed or has timed out. When a device restarts, it shall not reuse Network Connection IDs from previously opened connections until those connections have been closed or have timed out. A specific

.

connection ID shall not be reused so long as there is the possibility that packets with that connection ID are present in the network.” [Vol 2 Ed 1.17 3-3.7.1.1]

Two STC-defined connections are preferred: a point-to-point output connection and a multicast input connection.. If only one connection is available, it will be used for both test cases if T->O supports both point-to-point and multicast.

For the multicast case, both T->O and O->T Connection IDs are checked for non-reuse. For point-to-point case, only the O->T Connection ID is checked.

For each case, the test establishes an I/O connection and then performs a device reset; the user is prompted to power-cycle the device if the reset cannot be performed. After the reset wait time, the test attempts to open the same connection again. The Connection IDs determined by the target {see Vol 2 Ed 1.17 Table 3-3.2) should be different.

NOTE: This test assumes that the DUT uses a strategy similar to the Incarnation or Pseudo-Random techniques presented as informative material in Volume 2. Other techniques may also be determined to be effective that the current test cannot validate. Manual demonstration that the 3-3.7.1.1 requirement cited above is met will allow a failure of this test to be waived.

6.3.5) T->O Multicast Matching Rules test on the same network interface

a) T->O RPI mismatch test:

i. Open a multicast Class 1 I/O connection

ii. When above connection starts generating multicast I/O, open another multicast to the same port with the exactly same parameters except different T->O RPI.

Pass: The DUT returns General Status 0x01, Extended Status 0x0112.

Fail: Other responses

iii. Close opened connection(s).

b) T->O Network Connection Parameters size mismatch test:

i.Open a multicast Class 1 I/O connection

ii. When above connection starts generating multicast I/O, open another multicast to the same port with the exactly same parameters except different T->O connetion size.

Pass: The DUT returns General Status 0x01, Extended Status 0x0134 or 0x128 if the DUT cannot support the requested size.

Fail: Other responses

iii. Close opened connection(s).

c) T->O Network Connection Parameters Fixed/Variable mismatch test:

i. Open a multicast Class 1 I/O connection

ii. When above connection starts generating multicast I/O, open another multicast to the same port with the exactly same parameters except different T->O Fixed/Variable type.

Pass: The DUT returns General Status 0x01 and Extended Status 0x0135 or 0x120 if the DUT cannot support Variable type.

Fail: Other responses

iii. Close opened connection(s).

d) T->O Network Connection Parameters Priority mismatch test:

i. Open a multicast Class 1 I/O connection

ii. When above connection starts generating multicast I/O, open another multicast to the same port with the exactly same parameters except different T->O Priority.

Pass: The DUT returns General Status 0x01 amd Extended Status 0x0136 or 0x122 if the DUT cannot support the requested Priority.

Fail: Other responses

iii. Close opened connection(s).

e) Transport Class mismatch test:

i. Open a multicast Class 1 I/O connection

ii. When above connection starts generating multicast I/O, open another multicast to the same port with the exactly same parameters except Transport Class is 0.

Pass: The DUT returns General Status 0x01 and Extended Status 0x0137 or 0x11C if the DUT cannot support the requestsed Transport Class type.

Fail: Other responses

iii. Close opened connection(s).

f) T->O Production Trigger mismatch test:

i. Open a multicast Class 1 I/O connection

ii. When above connection starts generating multicast I/O, open another multicast to the same port with the exactly same parameters except different T->O Production Trigger.

Pass: The DUT returns General Status 0x01 and Extended Status 0x0138 or 0x11D if the DUT cannot support the requested Production Trigger type.

Fail: Other responses

iii. Close opened connection(s).

g) T->O Production Inhibit Time mismatch test if the DUT supports COS Production Trigger type:

   i. Open a multicast Class 1 I/O connection

   ii. When above connection starts generating multicast I/O, open another multicast to the same port with the exactly same parameters except different Production Inhibit Time.

   Pass: The DUT returns General Status 0x01 and Extended Status 0x0139.

   Fail: Other responses

   iii. Close opened connection(s).

h) T->O parameters totally match test:

   i. Open a multicast Class 1 I/O connection

   ii. When above connection starts generating multicast I/O, open another multicast to the same port with the exactly same parameters.

   Pass: The DUT successfully joined the existing connection with the same T->O Connection ID.

   Fail: The DUT returned error status code or create a new I/O connection.

   iii. Close opened connection(s).

6.4) GetConnectedData
There is no test for this service.

6.5) GetConnectionOwner
There is no test for this service.

GetConnectionOwner
There is no test for this service.

6.6) Class 3 Connected Explicit Messaging Tests
This test in only performed if the device indicates support for Class 3.

a) Class 3 PreConsumption Timer test
   1. Open a Class 3 connection
   2. Set a test timer for the greater of:
      i. 10000 ms plus 500 ms buffer or
      ii. 4 times RPI plus 500 ms buffer, where 4 times RPI is member variable (m_ReadWait), which is set by the user in GUI via Setup->MsgWait Timers, “Maximum Wait for All Msgs”.
   3. Wait for the test timer to expire.

4. Issue Get_Attribute_Single request to the TCP/IP object.

Pass: The Class 3 connection shall be timed out and no response shall be returned.

b) Class 3 Timeout Test
   1. Open a Class 3 connection
   2. Send Get_Attribute_Single to the TCP/IP object
   3. Receive Get_Attribute_Single response
   4. Wait 5*RPI for the Class 3 connection

   Pass: The Class 3 connection shall be timed out and no response shall be returned

c) Class 3 Duplicate 16-bit sequence count test
   1. Open a Class 3 connection
   2. Send a Get_Attribute_Single to Identity Object instance 1, attribute 1, with class 3 sequence count *n*
   3. Send same request from step 2 and expect the same response.
   4. Send a different request (Identity[1].GetAttributeSingle(6)) with the same sequence count from step 2

   Pass: the response shall match the response from step 2.

Part 7) I/O Test
If I/O parameters are configured for the Connection Manager object, then the following tests per set of I/O parameters are run.

If either the T->O or O->T connection size (including sequence count and header bytes, if applicable) is greater than 511 bytes, then LargeForwardOpen shall be used to establish the connections.

For any SockAddr Info item present in Forward_Open response, verify sin_family = AF_INET.

a) CIP I/O Injection test with Initial Sequence Number 0x80000000
   i. Open the I/O connection
   ii. Send I/O messages to the DUT until the first I/O message is received from the DUT. The O->T I/O message Sequence Number starts with 0x80000000.
   iii. After serveral O->T cycles, do a Sequence Number jump with gap > 16
   iv. Continue sending I/O messages to the DUT with incrementing Sequence Number by 1.

   Pass: The T->O connection remains open and produces data.

   Fail: The T->O connection finally times out (Connection Timeout Multiplier * RPI).

b) CIP I/O Injection Test with Initial Sequence Number 0xFFFFFFFE
   i. Open the I/O connection

ii. Send I/O messages to the DUT until the first I/O message is received from the DUT. The O->T I/O message Sequence Number starts with 0xFFFFFFFE.
iii. After serveral O->T cycles, do a Sequence Number jump with gap > 16
iv. Continue sending I/O messages to the DUT with incrementing Sequence Number by 1.
Pass: The T->O connection remains open and produces data.
Fail: The T->O connection finally times out (Connection Timeout Multiplier * RPI).

c) First Data Timeout (min 10 seconds)
After the Successful ForwardOpen is received, the test will fail the device if at any time during the next 10 seconds any two I/O UDP packets are more than 1500 ms apart, or if any received I/O UDP packets are invalid.

d) I/O 4xRPI Timeout
After the Successful ForwardOpen, a single heartbeat is sent to the DUT. The test will fail the device if during the next ten seconds more than five I/O packets are received or if and invalid I/O packet is received.

e) TCP close I/O test
i. Open the I/O connection
ii. Send I/O messages to the device until the first I/O message is received from the device
iii. Issue a *hard close* of the TCP connection that was used to create the I/O connection in step i.
iv. Continue sending I/O messages to the DUT.
Pass: The T->O connection remains open and produces data
Fail: The T->O connection times out
v. Create a TCP connection to the device, create a session and issue a ForwardClose for the I/O connection

f) TCP close I/O test (graceful shutdown)
i. Open the I/O connection
ii. Send I/O messages to the device until the first I/O message is received from the device
iii. Issue a *graceful close* of the TCP connection that was used to create the I/O connection in step i.
iv. Continue sending I/O messages to the DUT.
Pass: The T->O connection remains open and produces data.
Fail: The T->O connection times out.
v. Create a TCP connection to the device, create a session and issue a ForwardClose for the I/O connection

g) Encapsulation Sequence Number test
i. Open the I/O connection

ii. Send I/O messages to the DUT and increament the Encapsulation Sequence Number with one in each message until the first I/O message is received from the DUT

iii. Continue sending I/O messages to the DUT without incrementing the Encapsulation Sequence Number.
Pass: The T->O connection times out (after Connection Timeout Multiplier * RPI).
Fail: The T->O connection remains open and produces data.

iv. Create a TCP connection to the device, create a session and issue a ForwardClose for the I/O connection.

h) Out-of-Order I/O Packets test

i. Open the I/O connection

ii. Send I/O messages to the DUT with a non-zero Sequence Number, and increament the Encapsulation Sequence Number with one in each message until the first I/O message is received from the DUT. Remember this Sequence Number as the initial one.

iii. Continue sending I/O messages to the DUT with a random Encapsulation Sequence Number smaller than the initial one.
Pass: The T->O connection times out (after Connection Timeout Multiplier * RPI).
Fail: The T->O connection remains open and produces data.

iv. Create a TCP connection to the device, create a session and issue a ForwardClose for the I/O connection.

i) Reconnect a timeout I/O connection test

i. Open the I/O connection

ii. Make the I/O connection by only sending one O->T I/O message.

iii. After 4.4*O->T API, check if the I/O connection timeout.

Pass: The T->O connection times out and the underneath TCP connection is also closed.

Fail: The T->O connection remains open and produces data or the underneath TCP connection is still alive.

iv. Try to open the I/O connection again with the match Connection Triad (Connection Serial Number, Orignator Vendor ID and Originator Serial Number).

Pass: The I/O connection can be reopened correctly.

Fail: The DUT rejects to open the I/O connection with the matching Connection Triad even after 60 seconds.

v. Close the I/O connection.

Unregister the session at the conclusion of this test and leave the device in the On–Line state.

j) Point-Point T->O on non-default port test

This test verifies the the following statement in Vol 2 Ed 1.21 section 3-3.9.6:

"If the point-point consumer chooses a port number that is different than 0x08AE, then the point-point consumer shall send a Sockaddr Info item indicating the chosen port number."

i. Open an I/O connection with Point-Point T->O on a UDP port other than 2222.

Fail: The DUT is unable to open the connection on the requested port.

ii. Continuously receive T->O I/O for 10 second on the selected port.

Pass: The DUT generates I/O on the selected port.

Fail: The DUT does not generate I/O or still generate I/O on the default port 2222.

iii.Close the I/O connection.

k) Multicast and Point-Point Connection to the same connection path test

Reference of the test: Vol 1 Ed 3.20 section 3-6.3:

"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 connection."

i. If the DUT supports both Point to Point and Multicast connection, open a Multicast connection with Input Only Application Type.

ii. After receiving a Multicast I/O packet, open a Point-Point Connection with the same T->O Application Path.

Pass: The DUT successfully opened two connections to the same Application Path; or unable to open the Point-Point connection due to not enough resources.

Fail: The DUT was unable to open either connection due to non-resource related reason.

iii. Close both connections if opened.

4.6 Register Object Test x07

This section defines the conformance test for the Register object. This test is required when the Register object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Register object.
2) Instance attributes access rules for the Register object.
3) Common Service implementations for class and instance.
4) Object–specific and Reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
5.1) response when Direction attribute is set with an invalid value.
5.2) response when Size attribute is set with an invalid value.
5.3) response when Data attribute is set with too little or too much data, and Size is settable.
5.4) response when Data attribute is set with too little or too much data.
5.5) response when Data attribute is set with Direction = 0.
6) Object behavior accessible from the network.
6.1) behavior when data attribute is set with less data than the default.
6.2) behavior when data attribute is set with more data than the default.
6.3) behavior when attributes are set back to the default values.
6.4) Size and Data vlaues for RF and DC Power Generator Devices.

Procedure Definition

- Initialization
  The Register object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 2000 ms.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT, Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Bad_Flag (01)	[BOOL (0, 1)]
Direction (02)	[BOOL (0, 1)]
Size (03)	[UINT]
Data (04)	[Bit Array[Size]]
Undefined (05..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Bad_Flag (01)	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
Direction (02)	[Success_Response, Service_Not_Supported, Attribute_Not_Settable]
Size (03)	[Success_Response, Service_Not_Supported, Attribute_Not_Settable]
Data (04)	[Success_Response, Service_Not_Supported]
Undefined (05..99)	[Service_Not_Supported, Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] if no attributes supported
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (code)	[Expected Responses]
(00..13)	[Service_Not_Supported (x08)]
Get_Attribute_Single (14)	[requested value]

Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests. (Done only if the attribute is settable)

5.1) Set Direction Error

- Set_Attribute_Single, Direction = x0F
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.

5.2) Set Size Error

- Set_Attribute_Single, Size = 0
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.

5.3) Set Data Error Invalid size error

- Set_Attribute_Single, Direction = 1
- Set_Attribute_Single, Size = 10.
- Set_Attribute_Single, Data = x55.
  Pass: Error_Response 94 13 FF, Not_Enough_Data.
- Set_Attribute_Single, Data = x555555.
  Pass: Error_Response 94 15 FF, Too_Much_Data.
- Set_Attribute_Single, Size = original Size.

5.4) If Default Size is greater than 8 bits, Set Data Error Invalid sizes, Size = default

- Set_Attribute_Single, Data = Data value saved from step 2 - 1 byte.
  Pass: Error_Response 94 13 FF, Not_Enough_Data.
- Set_Attribute_Single, Data = Data value saved from step 2 + 1 byte (= x00).
  Pass: Error_Response 94 15 FF, Too_Much_Data.

5.5) Set Data when Direction = 0

- Set_Attribute_Single, Direction = 0.
- Set_Attribute_Single, Data = Data value saved from step 2.
  Pass: Error_Response 94 0E FF, Attribute_Not_Settable.

6) Behavior tests. If Size is not settable, skip the Behavior test.

- If Direction is settable, Set_Attribute_Single, Direction = 1.

6.1) Set Data Attribute less than default Size

- Set_Attribute_Single, Size = (default - 8), result Size must be > 0
- Set_Attribute_Single, Data = x55 for each byte, (Size - 8).
- Get_Attribute_Single, Data.
  Pass: Data = value from the Set_Attribute_Single request.

6.2) Set Data Attribute more than default Size

- Set_Attribute_Single, Size = (default + 8), result Size must be < 65535
- Set_Attribute_Single, Data = x55 for each byte, (Size + 8).
  Pass: Success_Response, or 94 15 FF, Too_Much_Data.
- Get_Attribute_Single, Data.
  Pass: Data = value from the last successful Set_Attribute_Single request.

6.3) Set Attributes back to default values saved from step 2.

- Use Get_Attribute_Single to read each of the four attributes.
  Pass: Each value must be the same as the original value.

6.4) Check Data for RF or DC Power Generator.

- Get_Attribute_Single, Instance 1, Size.
  Pass: Size = 8 (bits)
- Get_Attribute_Single, Instance 1, Data.
  Pass: No data in reserved bits 5, 6, 7 (or 3 for RF Power Generator)
- Get_Attribute_Single, Instance 2, Size.
  Pass: Size = 16 (bits)
- Get_Attribute_Single, Instance 2, Data.
  Pass: No data in reserved bits 14, 15

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.7 <u>Discrete Input Point Object Test x08</u>

This section defines the conformance test for the Discrete Input Point (DIP) object. This test is required when the Discrete Input Point object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Discrete Input Point object.

2) Instance attributes access rules for the Discrete Input Point object.

3) Common Service implementations for class and instance.

4) Object–specific and Reserved service implementations for class and instance.

5) Conformance when incorrect data is used to access attributes and services.

6) Object behavior accessible from the network.

6.1) Implementation of the Attributes_List attribute.

Procedure Definition

- Initialization
  The Discrete Input Point object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT (2)]
Max_Instance (02)	[UINT (1..65535), Attribute_Not_Supported]
Num_Instances (03)	[UINT , Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT (1..199), Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT (1..199), Attribute_Not_Supported]
Undefined (08..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	**[Expected values, Responses]**
Undefined (00)	[Attribute_Not_Supported (0x14)]
Num_Attributes (01)	[USINT (3..7), Attribute_Not_Supported]
Attributes_List (02)	[USINT[Num_Attributes], Attribute_Not_Supported]
Value (03)	[BOOL]
Status (04)	[BOOL, Attribute_Not_Supported]
Off_On_Delay (05)	[UINT, Attribute_Not_Supported]
On_Off_Delay (06)	[UINT, Attribute_Not_Supported]
Off_On_Cycles (07)	[UDINT, Attribute_Not_Supported]
Undefined (08..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	**[Expected Values, Responses]**
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Num_Attributes (01)	[Service_Not_Supported, Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes_List (02)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Value (03)	[Service_Not_Supported, Attribute_Not_Settable]
Status (04)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Off_On_Delay (05)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
On_Off_Delay (06)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Off_On_Cycles (07)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	**[Expected Responses]**
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response, requested value]
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
Set_Attributes_All (02)	[Success_Response, Service_Not_Supported]
(03..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response and requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Service_Not_Supported, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5.0) If Set_Attributes_All is implemented for the instance,

- Request a Set_Attributes_All using too much data.
  Pass: Error_Response 94 15 FF, Too_Much_Data.
- Request a Set_Attributes_All using too little data.
  Pass: Error_Response 94 13 FF, Not_Enough_Data.

6) Behavior Tests.

6.1) For each instance of the DIP object, if Instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.8 Discrete Output Point Object Test x09

This section defines the conformance test for the Discrete Output Point (DOP) object. This test is required when the Discrete Output Point object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Discrete Output Point object.
2) Instance attributes access rules for the Discrete Output Point object.
3) Common Service implementations for class and instance.
4) Object–specific service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
5.1) response when Set_Attributes_All is requested with invalid data.
5.2) response when Set_Attribute_Single is requested with invalid data.
6) Object behavior accessible from the network.
6.1) behavior of the Run and Idle commands before I/O connection is established.
6.2) default behavior of the Idle_State and Idle_Value attributes.
6.3) default behavior of the Fault_State and Fault_Value attributes.
6.4) behavior of Idle_Value attribute, if Settable.
6.5) behavior of Idle_State attribute, if Settable.
6.6) behavior of Fault_Value attribute, if Settable.
6.7) behavior of Fault_State attribute, if Settable.
6.8) value after I/O Connection is Deleted
6.9) Implementation of the Attributes_List attribute.

Procedure Definition

- Initialization
  The Discrete Output Point object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 5000ms.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attributes (06)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attributes (07)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Num_Attributes (01)	[USINT (1..199), Attribute_Not_Supported]
Attributes_List (02)	[USINT[Num_Attributes], Attribute_Not_Supported]
Value (03)	[BOOL (0, 1)]
Status (04)	[BOOL (0, 1), Attribute_Not_Supported]
Fault_State (05)	[BOOL (0, 1), Attribute_Not_Supported]
Fault_Value (06)	[BOOL (0, 1), Attribute_Not_Supported]
Idle_State (07)	[BOOL (0, 1), Attribute_Not_Supported]
Idle_Value (08)	[BOOL (0, 1), Attribute_Not_Supported]
Command (09)	[BOOL (0, 1), Attribute_Not_Supported]
Flash (10)	[BOOL (0, 1), Attribute_Not_Supported]
Flash_Rate (11)	[USINT (1..255), Attribute_Not_Supported]
Object_State (12)	[USINT (1..7, 255), Attribute_Not_Supported]
Test_Output_Mode_for_Safety (13)	[Attribute_Not_Supported] Note 1
Undefined (14..99)	[Attribute_Not_Supported]

Note 1: For Safety Devices, See CIP Safety PCTS (ODVA PUB00170) for test procedures

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Num_Attributes (01)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes_List (02)	[Attribute_Not_Settable, Attribute_Not_Supported]
Value (03)	[Success_Response]
Status (04)	[Attribute_Not_Settable, Attribute_Not_Supported]
Fault_State (05)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Fault_Value (06)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Idle_State (07)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Idle_Value (08)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Attribute Name (ID)	[Expected Values, Responses]
Command (09)	[Success_Response, Attribute_Not_Supported]
Flash (10)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Flash_Rate (11)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Object_State (12)	[Attribute_Not_Settable, Attribute_Not_Supported]
Test_Output_Mode_for_Safety (13)	[Attribute_Not_Supported] Note 1
Undefined (14..99)	[Attribute_Not_Supported]

Note 1: For Safety Devices, See CIP Safety PCTS (ODVA PUB00170) for test procedures

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value, Service_Not_Supported]
(15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00-49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	Note: If Number_of_Attributes = 0, Attributes_List is not returned. [(USINT(0..255), USINT[Num_Attributes], BOOL(0, 1), BOOL (0, 1), BOOL (0, 1), BOOL (0, 1), BOOL (0, 1), BOOL (0, 1), BOOL (0, 1), BOOL (0, 1), USINT (0..255), USINT (1..7, 255), Service_Not_Supported]
Set_Attributes_All (02)	[Success_Response, Service_Not_Supported]
(03..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable]
(17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100-255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Test.

5.1) If Set_Attributes_All is implemented for the instance,

- Request a Set_Attributes_All using too much data.
  Pass: Error_Response 94 15 FF, Too_Much_Data.
- Request a Set_Attributes_All using too little data.
  Pass: Error_Response 94 13 FF, Not_Enough_Data.

5.2) Request a Set_Attribute_Single, Instance attributes 3, 5..10, value = xFE.

Pass: For each attribute, Error_Response 94 09 FF, Invalid_Attribute_Value.

6) Behavior Test.

Note: This test description was written for both dynamically created and Predefined Master/Slave Connection Set connections. For Predefined Master/Slave Connections, the test uses a Group2 I/O message, depending on the I/O connection, to send Idle and Run commands. The Run I/O messages are the I/O data for the test and cause the DOP to transition the Run.

Note: If Instance attribute 9, Command, is implemented, set it as specified to put the DOP in the Idle or Ready state. Otherwise, ignore steps referring to the Command attribute. If the Command attribute is implemented, follow the Behavior Test without using the Command attribute. Then, repeat steps 6.1 through 6.4 using the Command Attribute to put the DOP in the Idle or Ready state.

Note: A Get_Attribute_Single request to the Command attribute must always return the value zero regardless of the value that is used with a Set_Attribute_Single request. This check is done for each Get_Attribute_Single request to the Command attribute but is only noted for the first two requests.

Note: If Instance attribute 9, Command, is not implemented and the device does not support the Predefined Master/Slave Connection Set, skip steps 6.1 through 6.4.

Note: Run command I/O messages contain the number of bytes specified by the Consumed Connection Size attribute and use non zero values unless otherwise noted.

6.0) Create/Allocate I/O Connection. If necessary, configure the I/O connection to access the DOP value attribute and set the connection Watchdog_Timeout_Action to transition to timed out.

Note: If the DOP is part of the I/O data (is controlled explicitly) start at step 6.1)

6.1e) Verify behavior for Explicit control

- Set Value Attribute = 1.
- Close the Explicit Message connection.
- Open the Explicit Message connection. Set EPR = 1 second
- Get the Value Attribute = 1.

Pass: Value = 1.

- Set a test timer for 5 seconds and wait for timer to expire.
- Open the Explicit Message connection. Set EPR = 1 second
- Get the Value Attribute = 1.
  Pass: Value = 1.
  skip to step 6.9

6.1) Run/Idle Test, I/O Messages

- Set Command Attribute = 0.
  Pass: Error_Response 94 0C FF, Object_State_Conflict.
- Send Idle I/O data command.
  Pass: No_Response.
- Set Command Attribute = 1.
  Pass: Error_Response 94 0C FF, Object_State_Conflict.
- Send Run I/O data command.
  Pass: No_Response.
- Request Set_Attribute_Single, Value = 1.
- Request Set_Attribute_Single, I/O Connection WatchDog_Timeout_Action = Time Out.

DOP transitions Available to Ready when connection transitions to Established state.

- Request Set_Attribute_Single, I/O Connection EPR = 500 milliseconds.
- For Dynamic I/O connections, request Apply_Attributes.

6.2) Idle Value Default Test

- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Set Command Attribute = 0. DOP transitions Ready to Idle.
- Get the Command Attribute
  Pass: Success_Response, Value = 0.
- Send Idle I/O data command.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.
- Set Command Attribute = 1. DOP transitions Idle to Ready.
- Request Get_Attribute_Single Command attribute.
  Pass: Success_Response, Value = 0.
- Send Run I/O data command, data = 1. DOP transitions Ready to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Send Run I/O data command, data = 0. DOP transitions Run to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.
- Request Set_Attribute_Single, Value = 1.
- Set Command Attribute = 0. DOP transitions Run to Idle.
- Send Idle I/O data command.
- Request Get_Attribute_Single, Value.

Pass: Success_Response, Value = 0.

- Set Command Attribute = 1. DOP transitions Idle to Ready.
- Send Run I/O data command, data = 1. DOP transitions Ready to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.

6.3) Fault Value Default Test

- Allow the I/O message connection to time out, transition Run to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.
- Reset or Create/Allocate the I/O Connection as done for step 6.1. transition Fault to Ready
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.
- Request Set_Attribute_Single, Value = 1.
- Allow the I/O connection to time out, transition Ready to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Reset or Create/Allocate the I/O Connection as done for step 6.1. transition Fault to Ready
- Set Command Attribute = 0. DOP transitions to Idle state.
- Send Idle I/O data command, transition Ready to Idle.
- Request Set_Attribute_Single, Value = 1.
- Allow the I/O connection to time out, transition Idle to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Reset or Create/Allocate the I/O Connection as done for step 6.1. transition Fault to Ready
- Set Command Attribute = 1. DOP transitions to Ready state.
- Send Run I/O data command, transition Ready to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.

6.4) If Idle_Value attribute is not settable, skip this step of the test.
Idle Value Test

- Request Set_Attribute_Single, Idle_Value = 1.
- Request Set_Attribute_Single, Value = 0.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.
- Set Command Attribute = 0. DOP transitions Run to Idle.
- Send Idle I/O data command.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Set Command Attribute = 1. DOP transitions Idle to Ready.
- Send Run I/O data command. DOP transitions Ready to Run.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 1.

- Request Set_Attribute_Single, Idle_Value = 0.

6.5) If the Idle_State attribute is not settable, skip this step of the test.
Idle State Test

- Request Set_Attribute_Single, Idle_State = 1. (Hold Last Value)
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 1.
- Set Command Attribute = 0. DOP transitions Run to Idle.
- Send Idle I/O data command. DOP transitions Run to Idle.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 1.
- Set Command Attribute = 1. DOP transitions Idle to Ready.
- Send Run I/O data command, data = 0. DOP transitions Ready to Run.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 0
- Request Set_Attribute_Single, Idle_State = 0.

6.6) If the Fault_Value attribute is not settable, skip this step of the test.
Fault Value Test

- Request Set_Attribute_Single, Fault_Value = 1.
- Request Set_Attribute_Single, Value = 0.
- Allow the I/O message connection to time out, transition Run to Fault.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 1.
- Reset or Create/Allocate the I/O Connection as done for step 6.1, transition Fault to Ready
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 1.
- Request Set_Attribute_Single, Fault_Value = 0.
- Send Run I/O data command. DOP transitions Ready to Run.

6.7) If the Fault_State attribute is not settable, skip this step of the test.
Fault State Test

- Request Set_Attribute_Single, Fault_State = 1 (Hold Last State).
- Request Set_Attribute_Single, Value = 1.
- Allow the I/O message connection to time out, transition Run to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Reset or Create/Allocate the I/O Connection as done for step 6.1, transition Fault to Ready
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Send Run I/O data command. DOP transitions Ready to Run.
- Request Set_Attribute_Single, Fault_State = 0.

6.8) Value after I/O Connection is Deleted

- Request Set_Attribute_Single, Value = 1.

- Release/Close the I/O Connection.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.

6.9) For each instance of the DOP object, if Instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.9 Analog Input Point Object Test x0A

This section defines the conformance test for the Analog Input Point (AIP) object. This test is required when the Analog Input Point object is implemented in the device.

Functional Description

This test verifies the:

1) class attribute access rules for the Analog Input Point object.

2) instance attribute access rules for the Analog Input Point object.

3) Common Service implementations for class and instance.

4) Object–specific and Reserved Service implementations for class and instance.

5) conformance when incorrect data is used to access attributes and services.

5.1) response when Set_Attributes_All is requested with invalid data.

6) object behavior accessible from the network.

6.1) implementation of the Attributes_List attribute.

Procedure Definition:

- Initialization
  The Analog Input Point object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT (2)]
Max_Instance (02)	[UINT (1..65535), Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..65535), Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT (1..199), Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT (1..199), Attribute_Not_Supported]
Undefined (08..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Num_Attributes (01)	[USINT 1..199, Attribute_Not_Supported]
Attributes_List (02)	[USINT[Num_Attributes], Attribute_Not_Supported]
Value (03)	[INT (-32768..32767), default or REAL (+/- 3.39 * e38), or USINT (0..255), or Type specified by Attribute 8]
Status (04)	[BOOL (0, 1), Attribute_Not_Supported]
Owner_Vendor (05)	[UINT , Attribute_Not_Supported]
Owner_Serial_No (06)	[(UDINT (0..4294967295), Attribute_Not_Supported]
Input_Range (07)	[USINT (0..5), Attribute_Not_Supported]
Value_Type (08)	[USINT (0..9, 100), Attribute_Not_Supported]
Undefined (09..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Num_Attributes (01)	[Service_Not_Supported, Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes_List (02)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Value (03)	[Service_Not_Supported, Attribute_Not_Settable]
Status (04)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Owner_Vendor (05)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Owner_Serial_No (06)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Input_Range (07)	[Success_Response, Service_Not_Supported, Attribute_Not_Supported]
Value_Type (08)	[Success_Response, Service_Not_Supported, Attribute_Not_Supported]
Undefined (09..99)	[Attribute_Not_Supported, Service_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response]

Service Name (Code)	[Expected Responses]
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
Set_Attributes_All (02)	[Success_Response, Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value, Attribute_Not_Supported]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable] Service_Not_Supported if Input_Range and Value_Type not implemented
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests

- Set_Attribute_Single, attribute 7 Input_Range = 7.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.
- Set_Attribute_Single, attribute 7 Input_Range = 15.

Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.

- Set_Attribute_Single, attribute 8, Value_Data_Type = 255.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.
- Set_Attribute_Single, attribute 10, Temp_Mode = 255.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.

5.1) If Set_Attributes_All is implemented for the instance,

- Request a Set_Attributes_All using too much data.
  Pass: Error_Response 94 15 FF, Too_Much_Data.
- Request a Set_Attributes_All using too little data.
  Pass: Error_Response 94 13 FF, Not_Enough_Data.

6) Behavior Tests.

6.1) For each instance of the AIP object, if Instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.10 Analog Output Point Object Test x0B

This section defines the conformance test for the Analog Output Point (AOP) object. This test is required when the Analog Output Point object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Analog Output Point object.
2) Instance attributes access rules for the Analog Output Point object.
3) Common Service implementations for class and instance.
4) Object–specific and Reserved Service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
5.1) response when Set_Attributes_All is requested with invalid data.
5.2) response when Set_Attribute_Single is requested with invalid data.
6) Object behavior accessible from the network.
6.1) behavior of the Run and Idle commands before I/O connection is established.
6.2) default behavior of the Idle_State and Idle_Value attributes.
6.3) default behavior of the Fault_State and Fault_Value attributes.
6.4) behavior of Idle_Value attribute, if Settable.
6.5) behavior of Idle_State attribute, if Settable.
6.6) behavior of Fault_Value attribute, if Settable.
6.7) behavior of Fault_State attribute, if Settable.
6.8) value after I/O Connection is Deleted
6.9) Implementation of the Attributes_List attribute.

Procedure Definition

- Initialization
  The Analog Output Point object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 5000ms.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT(1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT , Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.

Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[USINT (0..255), Attribute_Not_Supported]
Attributes_List (02)	[USINT[Num_Attributes], Attribute_Not_Supported]
Value (03)	[INT (-32768..32767), default or REAL (+/- 3.39 * e38), or USINT (0..255)] or Type depending on Attribute 8
Status (04)	[BOOL (0, 1), Attribute_Not_Supported]
Owner_Vendor_Id (05)	[UINT , Attribute_Not_Supported]
Owner_Serial_No (06)	[UDINT (0..4294967295), Attribute_Not_Supported]
Output_Range (07)	[USINT (0..5), Attribute_Not_Supported]
Value_Data_Type (08)	[USINT (0..9, 100), Attribute_Not_Supported]
Fault_State (09)	[USINT (0..3), Attribute_Not_Supported]
Idle_State (10)	[USINT (0..3), Attribute_Not_Supported]
Fault_Value (11)	[INT (-32768..32767) default. or Type depending on Attribute 8, Attribute_Not_Supported]
Idle_Value (12)	[INT (-32768..32767) default, or Type depending on Attribute 8, Attribute_Not_Supported]
Command (13)	[BOOL (0, 1), Attribute_Not_Supported]
Object_State (14)	[USINT (1..7, 255), Attribute_Not_Supported]
Undefined (15..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes_List (02)	[Attribute_Not_Settable, Attribute_Not_Supported]
Value (03)	[Success_Response]
Status (04)	[Attribute_Not_Settable, Attribute_Not_Supported]
Owner_Vendor_Id (05)	[Attribute_Not_Settable, Attribute_Not_Supported]
Owner_Serial_No (06)	[Attribute_Not_Settable, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Responses]
Output_Range (07)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Value_Type (08)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Fault_State (09)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Idle_State (10)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Fault_Value (11)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Idle_Value (12)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Command (13)	[Success_Response, Attribute_Not_Supported]
Object_State (14)	[Attribute_Not_Settable, Attribute_Not_Supported]
Undefined (15..99)	[Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] if no attributes are supported
(15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
Set_Attributes_All (02)	[Success_Response, Service_Not_Supported]
(03..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response]
(17..49)	[Service_Not_Supported]

4) Object Specific and Reserved Services Test

- Request each Object Specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object Specific Service request.

Service Name (Code)	[Expected Responses]
Object Specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object Specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object Specific Service request.

Service Name (Code)	[Expected Responses]
Object Specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) If Set_Attributes_All is implemented for the instance,

- Request a Set_Attributes_All using too much data.
  Pass: Error_Response 94 15 FF, Too_Much_Data.
- Request a Set_Attributes_All using too little data.
  Pass: Error_Response 94 13 FF, Not_Enough_Data.

5.2) For each attribute that is implemented with Set access,

- Request a Set_Attribute_Single service, Output_Range = 248.
- Request a Set_Attribute_Single service, Value_Data_Type = 240.
- Request a Set_Attribute_Single service, Fault_State = 248.
- Request a Set_Attribute_Single service, Idle_State = 248.
- Request a Set_Attribute_Single service, Command = 254.
  Pass: For each attribute, Error_Response 94 09 FF, Invalid_Attribute_Value.

6) Behavior Test

Note: This test description was written for both dynamically created and Predefined Master/Slave Connection Set connections. For Predefined Master/Slave Connections, the test uses a Group2 I/O message, depending on the I/O connection, to send Idle and Run commands. The Run I/O messages are the I/O data for the test and cause the AOP to transition the Run.

Note: If Instance attribute 13, Command, is implemented, set it as specified to put the AOP in the Idle or Ready state. Otherwise, ignore steps referring to the Command attribute. If the Command attribute is implemented, follow the Behavior Test without using the Command attribute. Then, repeat steps 6.1 through 6.7 using the Command Attribute to put the AOP in the Idle or Ready state.
Note: A Get_Attribute_Single request to the Command attribute must always return the value zero regardless of the value that is used with a Set_Attribute_Single request. This check is done for each Get_Attribute_Single request to the Command attribute but is only noted for the first two requests.
Note: If Instance attribute 13, Command, is not implemented and the device does not support the Predefined Master/Slave Connection Set, skip steps 6.1 through 6.7.
Note: Run command I/O messages contain the number of bytes specified by the Consumed

Connection Size attribute and use non zero values unless otherwise noted.

6.1) Create/Allocate I/O Connection. If necessary, configure the I/O connection to access the AOP value attribute and set the connection Watchdog_Timeout_Action to transition to timed out.

Note: If the AOP is part of the I/O data (is controlled explicitly) start at step 6.1)

6.1e) Verify behavior for Explicit control

- Set Value Attribute = Max Value.
- Close the Explicit Message connection.
- Open the Explicit Message connection. Set EPR = 1 second
- Get the Value Attribute = Max Value.
  Pass: Value = Max Value.
- Set a test timer for 5 seconds and wait for timer to expire.
- Open the Explicit Message connection. Set EPR = 1 second
- Get the Value Attribute = Max Value.
  Pass: Value = Max Value.
  skip to step 6.9

6.1) Run/Idle Test, I/O Messages

- Set Command Attribute = 0.
  Pass: Error_Response 94 0C FF, Object_State_Conflict.
- Send Idle I/O data command.
  Pass: No_Response.
- Set Command Attribute = 1.
  Pass: Error_Response 94 0C FF, Object_State_Conflict.
- Send Run I/O data command.
  Pass: No_Response.
- Request Set_Attribute_Single, Value = the average value for data type.
- Request Set_Attribute_Single, I/O Connection WatchDog_Timeout_Action = Time Out.
- Request Set_Attribute_Single, I/O Connection EPR = 500 milliseconds.
- For Dynamic I/O connections, request Apply_Attributes.

AOP transitions Available to Ready when connection transitions to Established state.

6.2) Idle Value Default Test

- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = the average value for data type.
- Set Command Attribute = 0. AOP transitions Ready to Idle.
- Get the Command Attribute
  Pass: Success_Response, Value = 0.
- Send Idle I/O data command.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = the average value for data type.
- Set Command Attribute = 1. AOP transitions Idle to Ready.
- Request Get_Attribute_Single Command attribute.
  Pass: Success_Response, Value = 0.

- Send Run I/O data command, data = average value for data type. AOP transitions Ready to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = average value for data type.
- Send Run I/O data command, data = average value for data type. AOP transitions Run to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = average value for data type.
- Request Set_Attribute_Single, Value = max value for data type.
- Set Command Attribute = 0. AOP transitions Run to Idle.
- Send Idle I/O data command.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = max value for data type.
- Set Command Attribute = 1. AOP transitions Idle to Ready.
- Send Run I/O data command, data = average value for data type. AOP transitions Ready to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = average value for data type.

6.3) Fault Value Default Test

- Allow the I/O message connection to time out, transition Run to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = average value for data type.
- Reset or Create/Allocate the I/O Connection as done for step 6.1. transition Fault to Ready
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = average value for data type.
- Request Set_Attribute_Single, Value = max value for data type.
- Allow the I/O connection to time out, transition Ready to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = max value for data type.
- Reset or Create/Allocate the I/O Connection as done for step 6.1. transition Fault to Ready
- Set Command Attribute = 0. AOP transitions to Idle state.
- Send Idle I/O data command, transition Ready to Idle.
- Request Set_Attribute_Single, Value = max value for data type.
- Allow the I/O connection to time out, transition Idle to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = max value for data type.
- Reset or Create/Allocate the I/O Connection as done for step 6.1. transition Fault to Ready
- Set Command Attribute = 1. AOP transitions to Ready state.
- Send Run I/O data command, transition Ready to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = max value for data type.

6.4) If Idle_Value attribute is not settable, skip this step of the test.
Idle Value Test

- Request Set_Attribute_Single, Idle_State = 3.
- Request Set_Attribute_Single, Idle_Value = minimum value for data type.

- Request Set_Attribute_Single, Value = average value for data type.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = average value for data type.
- Set Command Attribute = 0. AOP transitions Run to Idle.
- Send Idle I/O data command.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = minimum value for data type.
- Set Command Attribute = 1. AOP transitions Idle to Ready.
- Send Run I/O data command, data = average value for data type. AOP transitions Ready to Run.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = average value for data type.
- Request Set_Attribute_Single, Idle_Value = 0.

6.5) If the Idle_State attribute is not settable, skip this step of the test.

6.5.1) Idle State Test, Low Limit

- Request Set_Attribute_Single, Idle_State = 1. (Low Limit Value)
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = average value for data type.
- Set Command Attribute = 0. AOP transitions Run to Idle.
- Send Idle I/O data command. AOP transitions Run to Idle.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = minimum value for data type.
- Set Command Attribute = 1. AOP transitions Idle to Ready.
- Send Run I/O data command, data = average value for data type. AOP transitions Ready to Run.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = average value for data type.

6.5.2) Idle State Test, High Limit

- Request Set_Attribute_Single, Idle_State = 2. (High Limit Value)
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = average value for data type.
- Set Command Attribute = 0. AOP transitions Run to Idle.
- Send Idle I/O data command. AOP transitions Run to Idle.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = maximum value for data type.
- Set Command Attribute = 1. AOP transitions Idle to Ready.
- Send Run I/O data command, data = average value for data type. AOP transitions Ready to Run.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = average value for data type.
- Request Set_Attribute_Single, Idle_State = 0.

6.6) If the Fault_Value attribute is not settable, skip this step of the test.

Fault Value Test

- Request Set_Attribute_Single, Fault_State = 3.
- Request Set_Attribute_Single, Fault_Value = minimum value for data type + 100.

- Send Run I/O data command, data = average value for data type. AOP transitions Ready to Run.
- Allow the I/O message connection to time out, transition Run to Fault.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = minimum value for data type + 100.
- Reset or Create/Allocate the I/O Connection as done for step 6.1, transition Fault to Ready
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = minimum value for data type + 100.
- Send Run I/O data command, data = average value for data type. AOP transitions Ready to Run.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = average value for data type.
- Request Set_Attribute_Single, Fault_Value = 0.

6.7) If the Fault_State attribute is not settable, skip this step of the test.

6.7.1) Fault State Test, Low Limit

- Request Set_Attribute_Single, Fault_State = 1 (Low Limit).
- Request Set_Attribute_Single, Value = average value for data type.
- Allow the I/O message connection to time out, transition Run to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = minimum value for data type.
- Reset or Create/Allocate the I/O Connection as done for step 6.1, transition Fault to Ready
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = minimum value for data type.
- Send Run I/O data command, data = average value for data type. AOP transitions Ready to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = average value for data type.

6.7.2) Fault State Test, Low Limit

- Request Set_Attribute_Single, Fault_State = 2 (High Limit).
- Request Set_Attribute_Single, Value = average value for data type.
- Allow the I/O message connection to time out, transition Run to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = maximum value for data type.
- Reset or Create/Allocate the I/O Connection as done for step 6.1, transition Fault to Ready
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = maximum value for data type.
- Send Run I/O data command data = average for data type. AOP transitions Ready to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = average value for data type.
- Request Set_Attribute_Single, Fault_State = 0.

6.8) Value after I/O Connection is Deleted

- Request Set_Attribute_Single, Value = maximum value for data type.
- Release/Close the I/O Connection.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.

6.9) For each instance of the AOP object, if Instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.11 Presence Sensing Object Test x0E

This section defines the conformance test for the Presence Sensing Object. This test is required when the Presence Sensing Object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Presence Sensing object.

2) Instance attributes access rules for the Presence Sensing object.

3) Common Service implementations for class and instance.

4) Object-specific and Reserved Service implementations for class and instance.

5) Conformance when incorrect data is used to access attributes and services.

5.1) response when Operate_Mode is set with invalid value.

5.1) response when Target_Margin is set with invalid value.

5.2) response when Background_Margin is set with invalid value.

5.3) response when Min and Max Detect Distance are set with invalid values.

6) Object behavior accessible from the network.

6.1) Min_Detect_Distance, Max_Detect_Distance and Detect_Distance attribute values.

Procedure Definition

- Initialization
  The Presence Sensing Object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT , Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT (1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT (1..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]

Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (X14)]
Output (01)	[BOOL (0, 1)]
Num_Attributes (02)	[USINT(0..255), Attribute_Not_Supported]
Attributes_List (03)	[USINT[Num_Attributes], Attribute_Not_Supported]
Diagnostic (04)	[BOOL (0, 1), Attribute_Not_Supported]
On_Delay (05)	[UINT , Attribute_Not_Supported]
Off_Delay (06)	[UINT , Attribute_Not_Supported]
One_Shot_Delay (07)	[UINT , Attribute_Not_Supported]
Operate_Mode (08)	[BOOL (0, 1), Attribute_Not_Supported]
Sensitivity (09)	[USINT (0..255), Attribute_Not_Supported]
Target_Margin (10)	[USINT (1..255), Attribute_Not_Supported]
Background_Margin (11)	[USINT (0..100), Attribute_Not_Supported]
Min_Detect_Distance (12)	[UINT , Attribute_Not_Supported]
Max_Detect_Distance (13)	[UINT , Attribute_Not_Supported]
Detect_Distance (14)	[UINT , Attribute_Not_Supported]
Undefined (15..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Output (01)	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
Num_Attributes (02)	[Attribute_Not_Settable, Attribute_Not_Supported, Service_Not_Supported]
Attributes_List (03)	[Attribute_Not_Settable, Attribute_Not_Supported, Service_Not_Supported]
Diagnostic (04)	[Attribute_Not_Settable, Attribute_Not_Supported, Service_Not_Supported]
On_Delay (05)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported, Service_Not_Supported]
Off_Delay (06)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported, Service_Not_Supported]
One_Shot_Delay (07)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported, Service_Not_Supported]

Attribute Name (ID)	[Expected Responses]
Operate_Mode (08)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported, Service_Not_Supported]
Attribute Name (ID)	[Expected Responses]
Sensitivity (09)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported, Service_Not_Supported]
Target_Margin (10)	[Attribute_Not_Settable, Attribute_Not_Supported, Service_Not_Supported]
Background_Margin (11)	[Attribute_Not_Settable, Attribute_Not_Supported, Service_Not_Supported]
Min_Detect_Distance (12)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported, Service_Not_Supported]
Max_Detect_Distance (13)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported, Service_Not_Supported]
Detect_Distance (14)	[Attribute_Not_Settable, Attribute_Not_Supported, Service_Not_Supported]
Undefined (15..99)	[Attribute_Not_Supported, Service_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
(00..13)	[Service_Not_Supported (x08)]
Get_Attribute_Single (14)	[Success_Response, Service_Not_Supported If no attributes are supported]
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
(00..13)	[Service_Not_Supported (x08)]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response]
(17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]

Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) If Operate_Mode is settable, Set_Attribute_Single, Operate_Mode = 255,
Pass: Error_Response 94 09 FF, Invalid_Attribute_Value

5.2) If Target_Margin attribute (10) is settable.

- Set_Attribute_Single, Target_Margin = 0.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value

5.3) If Background_Margin attribute (11) is settable.

- Set_Attribute_Single, Background_Margin = 255.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value

5.4) If Min_Detect_Distance and Max_Detect_Distance attributes (12, 13) are settable.

- Set_Attribute_Single, Min_Detect_Distance = 100.
- Set_Attribute_Single, Max_Detect_Distance = 0.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value
- Set_Attribute_Single, Max_Detect_Distance = 0.
- Set_Attribute_Single, Min_Detect_Distance = 100.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value

6) Behavior Tests

6.1) Verify Min/Max/Detect distance attributes

- Get_Attribute_Single, Min_Detect_Distance, default value = 0.
- Get_Attribute_Single, Max_Detect_Distance = default value = 65535.
  Pass: Max_Detect_Distance > Min_Detect_Distance
- Get_Attribute_Single, Detect_Distance.

Pass: Detect_Distance < Max_Detect_Distance and Min_Detect_Distance < Detect_Distance
Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.12 Parameter Object Test x0F

This section defines the Parameter object conformance test. This test is required when the Parameter object is implemented in the device.

Functional Description

This test verifies the:

0) Object Descriptor and Instance implementation.

1) Class attributes access rules for the Parameter object.

2) Instance attributes access rules for the Parameter object.

3) Common Service implementations for class and instance.

4) Object–specific and Reserved Service implementations for class and instance.

5) Conformance when incorrect data is used to access attributes and services.

6) Object behavior accessible from the network.

6.1) Configuration assembly instance behavior.

6.2) Parameter_Value attribute Set_Attribute_Single access.

6.2.1) Parameter_Value attribute Set_Attribute_Single access with invalid data size.

6.3) Parameter_Value, Minimum_Value, and Maximum_Value attributes implementation.

6.3.1) Parameter_Value attribute Set_Attribute_Single access with invalid data value.

6.4) String attribute implementations.

6.5) Get_Enum_String service implementation.

6.6) Class Max_Instance value.

6.7) Behavior for International Strings.

Procedure Definition

- Initialization
  The Parameter object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

0) Object Description Verification

- Send a Get_Attribute_Single request to the Class Descriptor attribute.
- Extract the Parameter Instances and Full Attributes bits.
- Send a Get_Attribute_Single request to the Class Max_Instance attribute.
  Pass: If the Parameter Instances bit = 1, Success_Response and the Max_Instance value > 0.
- Save the Max_Instance value to be used later.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Attribute_Not_Supported]
Max_Instance (02)	[UINT]
Num_Instances (03)	[UINT , Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(9..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(6..199), Service_Not_Supported, Attribute_Not_Supported]
Parameter_Class_Descriptor (08)	[WORD (000000000000xxxx)]
Configuration_Assembly_Inst (09)	[UINT]
Language (10)	[USINT (0..7), Attribute_Not_Supported]
Undefined (11..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
(01 - 09)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Native_Language (10)	[Success_Response, Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Undefined (11..99)	[Service_Not_Supported, Attribute_Not_Supported]

2) Instance attribute access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Send a Get_Attribute_Single request to the Parameter Instance Descriptor attribute.
- Extract and save the value of each defined bit.
- Send a Get_Attribute_Single request to the Parameter Instance Data_Type attribute.
- Use this data to determine correct responses as shown in the Pass/Fail criteria tables
- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses] Full Parameter Instance
Undefined (00)	[Attribute_Not_Supported (x14)]
Parameter_Value (01)	Depends on Attributes 5, 6
Link_Path_Size (02)	[USINT (0..255)]
Link_Path (03)	[EPATH [Link_Path_Size] see Note 1 below]
Descriptor (04)	[WORD (0xxxxxxxxxxxxxxx)]
Data_Type (05)	[EPATH]
Data_Size (06)	[USINT] value determined by Data_Type
Parameter_Name_String (07)	[SHORT_STRING with max of 16 characters]
Units_String (08)	[SHORT_STRING with max of 4 characters]

Help_String (09)	[SHORT_STRING with max of 64 characters]
Minimum_Value (10)	[Same type as Parameter_Value (01)]
Maximum_Value (11)	[Same type as Parameter_Value (01)]
Default_Value (12)	[Same type as Parameter_Value (01)]
Scaling_Multiplier (13)	[UINT (1..65535)]
Scaling_Divisor (14)	[UINT (1..65535)]
Scaling_Base (15)	[UINT]
Scaling_Offset (16)	[INT (-32768..32767)]
Multiplier_Link (17)	[UINT (1..65535)]
Divisor_Link (18)	[UINT (1..65535)]
Base_Link (19)	[UINT]
Offset_Link (20)	[UINT]
Decimal_Precision (21)	[USINT (0..255)]
International_Parameter_Name (11)	[STRINGI, Attribute_Not_Supported]
International_Eng_Units (12)	[STRINGI, Attribute_Not_Supported]
International_HelpString (13)	[STRINGI, Attribute_Not_Supported]
Undefined (25..99)	[Attribute_Not_Supported]

Note 1: EPATH encoding is defined in the CIP Specification, Volume 1, Appendix C, Abstract Syntax Encoding for Segment Types. The expected values for an EPATH are as follows.

- Segment Type (bits 7, 6, 5) - 1, Logical Segment.
  Logical Format (bits 4, 3, 2) - 0, 1, 4; Class ID, Instance ID, Attribute ID.
  Logical Data Format (bits 1, 0) - 0, 1; 0 = USINT(1..255), 1 = UINT(1..65535).
  An EPATH must specify Class ID, and Instance ID. Attribute ID is optional.

Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses] Stub Parameter Instance
Undefined (00)	[Attribute_Not_Supported (x14)]
Parameter_Value (01)	[Success_Response] If Parameter_Value is not Read_Only [Attribute_Not_Settable(x0E)] If Parameter_Value is Read_Only
Link_Path_Size (02)	[Success_Response] If Link_Path is Settable [Attribute_Not_Settable] If Link_Path is not Settable
Link_Path (03)	Same as Link_Path_Size, #02
Descriptor (04)	[Attribute_Not_Settable]
Data_Type (05)	[Attribute_Not_Settable]
Data_Size (06)	[Attribute_Not_Settable]
Undefined (07..99)	[Attribute_Not_Supported]

Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses] Full Parameter Instance
Undefined (00)	[Attribute_Not_Supported (x14)]
Parameter_Value (01)	[Success_Response] If Parameter_Value is not Read_Only [Attribute_Not_Settable(x0E)] If Parameter_Value is Read_Only
Link_Path_Size (02)	[Success_Response] If Link_Path is Settable [Attribute_Not_Settable] If Link_Path is not Settable
Link_Path (03)	Same as Link_Path_Size, #02
(04..21)	[Attribute_Not_Settable]
(22..24)	[Attribute_Not_Settable, Attribute_Not_Supported]
Undefined (25..99)	[Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..04)	[Service_Not_Supported]
Reset (05)	[Success_Response, Service_Not_Supported]
(06..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable]
(17..20)	[Service_Not_Supported]
Restore (21)	[Success_Response, Service_Not_Supported]
Save (22)	[Success_Response, Service_Not_Supported]
(23..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Attribute_Not_Supported (x08)]
Get_Attributes_All (01)	Attributes 1, 10, 11, 12 specified by Instance Attributes 5 and 6, Data_Type and Data_Size, If Full Parameter Instances are Implemented, [Parameter_Value, USINT (0..255), PATH_ID, WORD (0000000000xxxxxx), USINT (1..26), USINT(1..255), SHORT_STRING, SHORT_STRING, SHORT_STRING, Minimum_Value, Maximum_Value, Default_Value, UINT, UINT , UINT , INT(32768..32767), UINT , UINT, UINT, UINT, USINT(0..255), STRINGI, STRINGI, STRINGI), Service_Not_Supported] If Stub Parameter Instances are Implemented, [Parameter_Value, USINT (0..255), PATH_ID, WORD (0000000000xxxxxx), USINT (1..3), USINT (1..255), possible vendor specific data) , Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	Attributes 1..6 [Success_Response] Attributes 7..21 [Success_Response] If Full Parameter Inst. Otherwise, [Attribute_Not_Supported]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Supported]
(17)	[Service_Not_Supported]
Get_Member (18)	[Success_Response, if STRINGI implemented Service_Not_Supported]
Reserved (19..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Services, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Services, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Get_Enum_String (75)	If bit 1 of Instance Attribute 4 = 1 [SHORT_STRING] for each value 0..Maximum_Value If bit 1 of Instance Attribute 4 = 0, Service_Not_Supported(x08)]
Object–specific Codes (76..99)	[Service_Not_Supported]

- Request each of the Reserved Services, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error tests.

- Send a Set_Attribute_Single request to Instance attribute 6, Link_Path, with invalid path data.
  Note: The test is only done if fragmented messaging is implemented.
  Pass: The expected response from table below for the Set_Attribute_Single request.

If the Link_Path is Settable	If Link_Path is NOT Settable and Fragmented Messaging
[Invalid_Attribute_Value (x09)]	[Attribute_Not_Settable (x0E)]

5.1) Service Test, if implemented.

- Request a Get_Member, Destination_List, MemberId = 255.
  Pass: Error_Response, 94 20 FF, Invalid Parameter

6) Behavior Tests.

6.1) Configuration Assembly Instance verification
Bit 0 of the Parameter Class Descriptor attribute is the Parameter Instances bit. If it = 0, there must be a Configuration Assembly Instance and it's ID specified by Parameter Class attribute 9.

- Send a Get_Attribute_Single request to the Configuration Assembly Instance attribute.
  Pass: The expected response from table below for the Set_Attribute_Single request.

If Parameter_Instances bit = 0	If Parameter_Instances bit = 1
[Success_Response and Id > 0]	[Success_Response and Id = 0]

- If Id > 0, send a Get_Attribute_Single request to the Configuration Assembly Data attribute.
  Pass: Success_Response.

6.2) Check Parameter Instance 1 through Parameter Max_Instance.

- Verify the Parameter Instance Id <= Parameter Max_Instance attribute value.
- Send a Get_Attribute_Single request to the Parameter Instance Descriptor attribute.
- Extract and save the value of each defined bit.
- Send a Get_Attribute_Single request to the Parameter Instance Data_Type attribute.
- Use this data to determine correct responses as shown in the Pass/Fail criteria tables.

6.2.1) Send a Get_Attribute_Single request to the Parameter_Value attribute.

- Compare number of bytes in Value to the size specified by Data_Type.
  Pass: Value data size = Data_Type size defined.
- Send a Set_Attribute_Single request to the Parameter_Value attribute.
  Pass: The expected response from table below for the Set_Attribute_Single request.

If Descriptor Read_Only bit = 0	If Descriptor Read_Only bit = 1
[Success_Response]	[Attribute_Not_Supported (x14), Service_Not_Supported (x08)]

6.2.1) Set Value with Invalid sizes. If Descriptor Read_Only bit = 0 (Parameter_Value is settable).

- Request a Set_Attribute_Single using too much data for the Parameter_Value Type and Size.
  Pass: Error_Response 94 15 FF, Too_Much_Data.
- Request a Set_Attribute_Single using too little data for the Parameter_Value Type and Size.
  Pass: Error_Response 94 13 FF, Not_Enough_Data.

6.3) Verify Min/Max/Value. If the attributes are implemented, otherwise, skip to step 6.4.

Note: If Get_Enum_String is implemented, Minimum_Value and Maximum_Value are required.

- Send a Get_Attribute_Single request to the Parameter_Value attribute.
- Send a Get_Attribute_Single request to the Minimum_Value attribute.
- Send a Get_Attribute_Single request to the Maximum_Value attribute.
- Send a Get_Attribute_Single request to the Default_Value attribute, for a full parameter.
  Pass: Success_Response and the value for each attribute.

  Pass: Minimum_Value <= Parameter_Value <= Maximum_Value

  Pass: Minimum_Value <= Default_Value <= Maximum_Value

6.3.1) Set Value range check

Note: Skip this step unless Parameter_Value is settable and the Data_Type range allows the test. For example, if Data_Type = UINT and Minimum_Value > 0, then set Parameter_Value < Minimum_Value. If Maximum_Value < 65535, then set Parameter_Value > Maximum_Value.

- Request a Set_Attribute_Single, Parameter_Value = Minimum_Value - 1.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.
- Request a Set_Attribute_Single, Parameter_Value = Minimum_Value + 1.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.

6.4) Check String Attributes, if the Name, Units or Help String attributes are implemented

- Send a Get_Attribute_Single request to the Parameter_Name_String attribute.
- Send a Get_Attribute_Single request to the Units_String attribute.
- Send a Get_Attribute_Single request to the Help_String attribute.
  Pass: Success_Response and the SHORT_STRING value for each attribute.

  Pass: The string is the correct length and each character in the string is a printable character.

6.5) Get_Enum_String service verification.

- Send a Get_Attribute_Single request to the Minimum_Value attribute.
- Send a Get_Attribute_Single request to the Maximum_Value attribute.
- Request a Get_Enum_String service for each value from Minimum_Value to Maximum_Value.
  Pass: Success_Response and the string value for each value.

  Pass: The string is the correct length and each character in the string is a printable character.
- Request a Get_Enum_String service for Max_Instance plus one.
  Pass: Error_Response 94 20 FF, Invalid_Parameter_Value.

6.6) Verify Parameter Instances = Class Max_Instance.

- Compare the last Instance Id to value of Class Max_Instance.
  Pass: Instance Id = Class Max_Instance

6.7) Check International Strings.

- request a Get_Attribute_Single, Identity, instance 1, Supported_Language_List.
  Pass: Success_Response

- determine the number of identifiers
- request a Get_Member service, member = 0 for each International String attribute
  Pass: Success_Response, valid STRINGI data for active language
- request a Get_Member service, for each member, for each International String attribute
  Pass: Success_Response, valid STRINGI data for requested language
  Pass: Error_Response, 94 02 FF, Resource_Unavailable

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.13 Parameter Group Object Test x10

This section defines the conformance test for the Parameter Group object. This test is required when the Parameter Group object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Parameter Group object.

2) Instance attributes access rules for the Parameter Group object.

3) Common Service implementations for class and instance.

4) Object–specific service implementations for class and instance.

5) Conformance when incorrect data is used to access attributes and services.

6) Object behavior accessible from the network.

6.1) Name_String attribute.

6.2) Number_Of_Members_in_Group attribute.

6.3) Each Parameter Instance member of the Parameter Group.

6.4) Response for an invalid Parameter Group instance access.

6.5) Behavior for International Strings.

Procedure Definition

- Initialization
  The Parameter Group object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

0) Configure Test for Object Implementation

- Request a Get_Attribute_Single service, Parameter Group class, Max_Instance attribute.
- Save the value of the Parameter Group Max_Instance attribute.
- Request a Get_Attribute_Single service, Parameter class, Max_Instance attribute.
- Save the value of the Parameter Max_Instance attribute.
- Request a Get_Attribute_Single service, Parameter Group instance attribute 2.
- Use the value of Number_Of_Members_in_Group to evaluate each attribute access test.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1, 2), Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535)]
Num_Instances (03)	[UINT , Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(8..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(3..199), Service_Not_Supported, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Native_Language (08)	[USINT(0..7) (only if Revision = 1), Attribute_Not_Supported]
Undefined (09..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Attribute_Not_Supported (x14), Service_Not_Supported (x08)]
(01..07)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported,]
Native_Language (08)	[Success_Response, Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Undefined (09..99)	[Attribute_Not_Supported, Service_Not_Supported]

2) Instance attribute access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

For Revision = 1	
Attribute Name (ID)	**[Expected values, Responses]**
Undefined (00)	[Attribute_Not_Supported (x14)]
Group_Name_String (01)	[SHORT_STRING] Maximum 16 Characters
Number_Of_Members_in_Group (02)	[UINT (1..65535)] value = Group_Size
Parameter_Instance_ID_1 (03)	[UINT (1..65535)]
Parameter_Instance_ID_2 (04)	[UINT (1..65535)]
Parameter_Instance_ID Group_Size	[UINT (1..65535)]
Undefined ((Group_Size + 1)..99)	[Attribute_Not_Supported]

For Revision > 1	
Attribute Name (ID)	**[Expected values, Responses]**
Undefined (00)	[Attribute_Not_Supported (x14)]
Group_Name_String (01)	[SHORT_STRING] Maximum 16 Characters
Number_Of_Members_in_Group (02)	[UINT (1..65535)] value = Group_Size
International_Group_Name (03)	[STRINGI]
Reserved (4..15)	[Attribute_Not_Supported]
Parameter_Instance_ID_1 (16)	[UINT (1..65535)]
Parameter_Instance_ID Group_Size	[UINT (1..65535)]
Undefined ((Group_Size + 1)..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected values, Responses]
(00..99)	[Service_Not_Supported (x08)]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01) Revision = 1	[(SHORT_STRING, UINT(Members), UINT[Members]), Service_Not_Supported]
Get_Attributes_All (01) Revision > 1	[(SHORT_STRING, UINT(Members), STRINGI, UINT[Members]), Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response] If Attribute 8 is Settable [Service_Not_Supported]
(17..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[SHORT_STRING, UINT (1..65535), UINT (1..65535), UINT (1..65535), For as many Instances as are in Group), Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Get_Member (18)	[Success_Response, if STRINGI implemented Service_Not_Supported]
(17, 18..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Services, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Services, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error tests.

- If class attribute 8, Language, is settable,
  Request a Set_Attribute_Single service, Instance 0, Attribute 8 = 255
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value

5.3) Service Test, if implemented.

- Request a Get_Member, Destination_List, MemberId = 255.
  Pass: Error_Response, 94 20 FF, Invalid Parameter

6) Behavior tests.

Check Parameter Group Instance 1 through Max_Instance.

Note: The Parameter Group Object test uses the Parameter Object Max_Instance class attribute to verify that each instance stored in the Parameter Group Object is a valid Parameter Object Id.

6.1) Check Parameter Group Name.

- Request a Get_Attribute_Single service, Parameter Group, Group_Name_String.
  Pass: Success_Response and the SHORT_STRING Name value. All characters printable.

6.2) Check Number_Of_Members_in_Group.

- Request a Get_Attribute_Single service, Parameter Group, Number_Of_Members_in_Group.
  Pass: Success_Response, value > 0 and <= the Parameter Max_Instance Id.

6.3) Check Parameter Instances of Parameter Group

Note: Start reading at attribute 3 (the first Parameter Instance ID in the Group). Read the number of attributes specified by Parameter Group, Number_Of_Members_in_Group attribute.

- Request a Get_Attribute_Single service, for each Parameter Instance in the Parameter Group.
- Request a Get_Attribute_Single service, Parameter Instance, Descriptor.
  Pass: Success_Response and the Parameter Instance Descriptor data.
- Request a Get_Attribute_Single service, Parameter Group Instance,
  Number_Of_Members_in_Group + 1.
  Pass: Error_Response 94 14 FF, Attribute_Not_Supported

6.4) Request a Get_Attribute_Single service, Parameter Group Max_Instance + 1, attribute 2.

Note: Read Number_Of_Members_in_Group + 1 to verify the end of the attributes.

Pass: Error_Response 94 16 FF, Object_Does_Not_Exist

6.5) Check International Strings.

- Request a Get_Attribute_Single, Identity, instance 1, Supported_Language_List.
  Pass: Success_Response
- Determine the number of identifiers
- Request a Get_Member service, member = 0, International_Grorup_Name attribute
  Pass: Success_Response, valid STRINGI data for active language
- Request a Get_Member service for each International_Grorup_Name member
  Pass: Success_Response, valid STRINGI data for requested language
  Pass: Error_Response, 94 02 FF, Resource_Unavailable

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.14 Group Object Test x12

This section defines the conformance test for the Group object. This test is required when the Group object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Group object.

2) Instance attributes access rules for the Group object.

3) Common Service implementations for class and instance.

4) Object–specific service implementations for class and instance.

5) Conformance when incorrect data is used to access attributes and services.

6) Object behavior accessible from the network.

6.1) Implementation of the Attributes_List attribute.

6.2) Implementation of the Bindings attribute.

Procedure Definition

- Initialization
  The Group object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT, Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(6..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(3..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[USINT (3..7), Attribute_Not_Supported]
Attributes_List (02)	[USINT[Number_of_Attributes], Attribute_Not_Supported]
Number_of_Bound_Instances (03)	[USINT, Attribute_Not_Supported]
Binding, Class/Instance List (04)	[(UINT, UINT)[Bound_Instances], Attribute_Not_Supported]
Status (05)	[BOOL, Attribute_Not_Supported]
Owner_Vendor (06)	[UINT, Attribute_Not_Supported]
Owner_Serial_No (07)	[(UDINT, Attribute_Not_Supported]
Undefined (08..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
All Attributes (00..99)	[Service_Not_Supported (x14)]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Service_Success, [Service_Not_Supported, If no attributes are supported]
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	**Note**: If either Number_of_Attributes or Number_of_Bound_Instances = 0, the corresponding list attribute is not returned [USINT(0..106), USINT[Number_of_Attributes], USINT (1..255), (UINT(1..65535), UINT(1..65535)) [Number_of_Bound_Instances], BOOL, UINT, UDINT, Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value, Service_Not_Supported]
Reserved (15..49)	[Service_Not_Supported]

4) Object Specific and Reserved Services Test

- Request each Object Specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object Specific Service request.

Service Name (Code)	[Expected Responses]
Object Specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, No_Response]

- Request each Object Specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object Specific Service request.

Service Name (Code)	[Expected Responses]
Object Specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, No_Response]

5) Error Tests

No additional testing is required.

6) Behavior Tests

6.1) For each Group object instance, if instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.

- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

6.2) For each Group object instance, if instance attribute 4, Binding, is implemented,

- Request a Get_Attribute_Single, Number_of_Bound_Instances.
- Request a Get_Attribute_Single, Binding.
  Pass: Service_Success with a valid Class Id = x08, x09, x0A, x0B, x1D, x1E, x1F, x20, x21, x22 and valid Instance Id.
- Request a Get_Attribute_Single, attribute 3, for each instance in the Binding list.
  Pass: Success_Response or Error_Response that is **NOT** 94 16 FF, Object_Does_Not_Exist

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.15 Discrete Input Group Object Test x1D

This section defines the conformance test for the Discrete Input Group (DIG) object. This test is required when the Discrete Input Group object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Discrete Input Group object.
2) Instance attributes access rules for the Discrete Input Group object.
3) Common Service implementations for class and instance.
4) Object–specific service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
6) Object behavior accessible from the network.
6.1) Implementation of the Attributes_List attribute.
6.2) Implementation of the Bindings attribute.

Procedure Definition

- Initialization
  The Discrete Input Group object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14), Service_Not_Supported (x08)]
Revision (01)	[UINT (1), Attribute_Not_Supported, Service_Not_Supported]
Max_Instance (02)	[UINT , Attribute_Not_Supported, Service_Not_Supported]
Num_Instances (03)	[UINT , Attribute_Not_Supported, Service_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Attribute_Not_Supported, Service_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Attribute_Not_Supported, Service_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(1..199), Attribute_Not_Supported, Service_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(1..199), Attribute_Not_Supported, Service_Not_Supported]
Undefined (08..99)	[Attribute_Not_Supported, Service_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 0 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Num_Attributes (01)	[USINT (1..106), Attribute_Not_Supported]
Attributes_List (02)	[USINT[Num_Attributes], Attribute_Not_Supported]
Number_of_Bound_Instances (03)	[USINT, Attribute_Not_Supported]
Binding, Instance List (04)	[UINT[Number_of_Bound_Instances], Attribute_Not_Supported]
Status (05)	[BOOL, Attribute_Not_Supported]
Off_On_Delay (06)	[UINT, Attribute_Not_Supported]
On_Off_Delay (07)	[UINT, Attribute_Not_Supported]
Undefined (08..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Num_Attributes (01)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes_List (02)	[Attribute_Not_Settable, Attribute_Not_Supported]
Number_of_Bound_Instances (03)	[Attribute_Not_Settable, Attribute_Not_Supported]
Binding, DIP Instance Ids (04)	[Attribute_Not_Settable, Attribute_Not_Supported]
Status (05)	[Attribute_Not_Settable, Attribute_Not_Supported]
Off_On_Delay (06)	[Service_Success, Attribute_Not_Settable, Attribute_Not_Supported]
On_Off_Delay (07)	[Service_Success, Attribute_Not_Settable, Attribute_Not_Supported]
Undefined (08..99)	[Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] If no attributes are supported
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	**Note:** If either Number_of_Attributes or Number_of_Bound_Instances = 0,

	the corresponding list attribute is not returned [USINT(0..106), USINT[Number_of_Attributes], USINT, UINT[Number_of_Bound_Instances], BOOL, UINT, UINT, Service_Not_Supported]
Set_Attributes_All (02)	[Service_Success, Service_Not_Supported]
(03..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value, Service_Not_Supported]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Service_Success, Service_Not_Supported, Attribute_Not_Settable]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests

If Set_Attributes_All is implemented for the instance,

- Request a Set_Attributes_All using too much data.
  Pass: Error_Response 94 15 FF, Too_Much_Data.
- Request a Set_Attributes_All using too little data.
  Pass: Error_Response 94 13 FF, Not_Enough_Data

6) Behavior tests.

6.1) For each DIG object instance, if instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

6.2) For each DIG object instance, if instance attribute 4, Binding, is implemented,

- Request a Get_Attribute_Single, Number_of_Bound_Instances.
- Request a Get_Attribute_Single, Binding.
  Pass: Success_Response with a valid DIP Instance Id.
- Request a Get_Attribute_Single, Value, for each DIP instance in the Binding list.
  Pass: Success_Response, value for each DIP instance.

6.2.1) Check Attributes of Bound Instances

- Request a Get_Attribute_Single, Off_On_delay and On_Off_Delay.
- Request a Set_Attribute_Single for each attribute of the bound instance(s).
  Pass: Error_Response 94 0E FF, attribute not settable (or 14, attribute not implemented).
- Request a Get_Attribute_Single each attribute, for each DIP instance.
  Pass: Success_Response, point value = group value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.16 Discrete Output Group Object Test x1E

This section defines the conformance test for the Discrete Output Group (DOG) object. This test is required when the Discrete Output Group object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Discrete Output Group object.

2) Instance attributes access rules for the Discrete Output Group object.

3) Common Service implementations for class and instance.

4) Object–specific service implementations for class and instance.

5) Conformance when incorrect data is used to access attributes and services.

5.1) response when Set_Attributes_All is requested with invalid data.

5.2) response when Set_Attribute_Single is requested with invalid data.

6) Object behavior accessible from the network.

6.1) behavior of the Run and Idle commands before I/O connection is established.

6.2) default behavior of the Idle_State and Idle_Value attributes.

6.3) default behavior of the Fault_State and Fault_Value attributes.

6.4) behavior of Idle_Value attribute, if Settable.

6.5) behavior of Idle_State attribute, if Settable.

6.6) behavior of Fault_Value attribute, if Settable.

6.7) behavior of Fault_State attribute, if Settable.

6.8) value after I/O Connection is Deleted

6.9) implementation of the Attributes_List attribute.

6.10) implementation of the Bindings attribute.

Procedure Definition

- Initialization
  The DOG object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (0..65535), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(6..199), Service_Not_Supported, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[USINT, Attribute_Not_Supported]
Attributes_List (02)	[USINT[Num_Attributes], Attribute_Not_Supported]
Number_of_Bound_Instances (03)	[USINT, Attribute_Not_Supported]
Binding, Instance List (04)	[UINT[Bound_Instances], Attribute_Not_Supported]
Status (05)	[BOOL, Attribute_Not_Supported]
Command (06)	[BOOL]
Fault_State (07)	[BOOL, Attribute_Not_Supported]
Fault_Value (08)	[BOOL, Attribute_Not_Supported]
Idle_State (09)	[BOOL, Attribute_Not_Supported]
Idle_Value (10)	[BOOL, Attribute_Not_Supported]
Undefined (11..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Attribute_Not_Supported (x14),]
Number_of_Attributes (01)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes_List (02)	[Attribute_Not_Settable, Attribute_Not_Supported]
Number_of_Bound_Instances (03)	[Attribute_Not_Settable, Attribute_Not_Supported]
Binding, Instance List (04)	[Attribute_Not_Settable, Attribute_Not_Supported]
Status (05)	[Attribute_Not_Settable, Attribute_Not_Supported]
Command (06)	[Service_Success]

Attribute Name (ID)	[Expected Responses]
Fault_State (07)	[Service_Success, Attribute_Not_Settable, Attribute_Not_Supported]
Fault_Value (08)	[Service_Success, Attribute_Not_Settable, Attribute_Not_Supported]
Idle_State (09)	[Service_Success, Attribute_Not_Settable, Attribute_Not_Supported]
Idle_Value (10)	[Service_Success, Attribute_Not_Settable, Attribute_Not_Supported]
Undefined (11..99)	[Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Service_Success, Service_Not_Supported, If no attributes are supported]
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	**Note:** If either Number_of_Attributes or Number_of_Bound_Instances = 0, the corresponding list attribute is not returned. [(USINT, USINT[Number_of_Attributes], USINT, UINT[Number_of_Bound_Instances], BOOL, BOOL, BOOL, BOOL, BOOL, BOOL), Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Service_Success, Attribute_Not_Settable (x0E)]
(17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]

Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Test.

5.1) If Set_Attributes_All is implemented for the instance and no vendor attributes are implemented,

- Request a Set_Attributes_All using too much data.
  Pass: Error_Response 94 15 FF, Too_Much_Data.
- Request a Set_Attributes_All using too little data.
  Pass: Error_Response 94 13 FF, Not_Enough_Data.

5.2)

- Request a Set_Attribute_Single, Command = 254.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.
- Request a Set_Attribute_Single, Fault_State = 254.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.
- Request a Set_Attribute_Single, Fault_Value = 254.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.
- Request a Set_Attribute_Single, Idle_State = 254.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.
- Request a Set_Attribute_Single, Idle_Value = 254.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.

6) Behavior Test.

Note: This test description was written for both dynamically created and Predefined Master/Slave Connection Set connections. For Predefined Master/Slave Connections, the test uses a Group2 I/O message, depending on the I/O connection, to send Idle and Run commands. The Run I/O messages are the I/O data for the test and cause each DOP in the DOG to transition to Run.

Note: Set the Command attribute as specified to put the DOPs in the Idle or Ready state. Follow the Behavior Test without using the Command attribute. Then, repeat steps 6.1 through 6.7 using the Command Attribute to put the DOP in the Idle or Ready state.

Note: A Get_Attribute_Single request to the Command attribute must always return the value zero regardless of the value that is used with a Set_Attribute_Single request. This check is done for each Get_Attribute_Single request to the Command attribute but is only noted for the first two requests.

Note: Run command I/O messages contain the number of bytes specified by the Consumed Connection Size attribute and use non zero values unless otherwise noted.

6.1) Create/Allocate I/O Connection. If necessary, configure the I/O connection to access grouped DOP value attributes. Set the connection Watchdog_Timeout_Action to transition to timed out.

- Set Command Attribute = 0.
  Pass: Error_Response 94 0C FF, Object_State_Conflict.
- Send Idle I/O data command.
  Pass: No_Response.
- Set Command Attribute = 1.
  Pass: Error_Response 94 0C FF, Object_State_Conflict.
- Send Run I/O data command.
  Pass: No_Response.
- Request Set_Attribute_Single, Value = 1.
- Request Set_Attribute_Single, I/O Connection WatchDog_Timeout_Action = Time Out.

DOPs transitions Available to Ready when connection transitions to Established state.

- Request Set_Attribute_Single, I/O Connection EPR = 500 milliseconds.
- For Dynamic I/O connections, request Apply_Attributes.

6.2) Idle Value Default Test

- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Set Command Attribute = 0. DOPs transitions Ready to Idle.
- Get the Command Attribute
  Pass: Success_Response, Value = 0.
- Send Idle I/O data command.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.
- Set Command Attribute = 1. DOPs transitions Idle to Ready.
- Request Get_Attribute_Single Command attribute.
  Pass: Success_Response, Value = 0.
- Send Run I/O data command, data = 1. DOPs transitions Ready to Run.
- Request Get_Attribute_Single, Value.

Pass: Success_Response, Value = 1.

- Send Run I/O data command, data = 0. DOPs transitions Run to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.
- Request Set_Attribute_Single, Value = 1.
- Set Command Attribute = 0. DOPs transitions Run to Idle.
- Send Idle I/O data command.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.
- Set Command Attribute = 1. DOPs transitions Idle to Ready.
- Send Run I/O data command, data = 1. DOPs transitions Ready to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.

6.3) Fault Value Default Test

- Allow the I/O message connection to time out, transition Run to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.
- Reset or Create/Allocate the I/O Connection as done for step 6.1. transition Fault to Ready
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.
- Request Set_Attribute_Single, Value = 1.
- Allow the I/O connection to time out, transition Ready to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Reset or Create/Allocate the I/O Connection as done for step 6.1. transition Fault to Ready
- Set Command Attribute = 0. DOPs transitions to Idle state.
- Send Idle I/O data command, transition Ready to Idle.
- Request Set_Attribute_Single, Value = 1.
- Allow the I/O connection to time out, transition Idle to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Reset or Create/Allocate the I/O Connection as done for step 6.1. transition Fault to Ready
- Set Command Attribute = 1. DOPs transitions to Ready state.
- Send Run I/O data command, transition Ready to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.

6.4) If Idle_Value attribute is not settable, skip this step of the test.
Idle Value Test

- Request Set_Attribute_Single, Idle_Value = 1.
- Request Set_Attribute_Single, Value = 0.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.

- Set Command Attribute = 0. DOPs transitions Run to Idle.
- Send Idle I/O data command.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Set Command Attribute = 1. DOPs transitions Idle to Ready.
- Send Run I/O data command. DOPs transitions Ready to Run.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 1.
- Request Set_Attribute_Single, Idle_Value = 0.

6.5) If the Idle_State attribute is not settable, skip this step of the test.

Idle State Test

- Request Set_Attribute_Single, Idle_State = 1. (Hold Last Value)
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 1.
- Set Command Attribute = 0. DOPs transitions Run to Idle.
- Send Idle I/O data command. DOPs transitions Run to Idle.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 1.
- Set Command Attribute = 1. DOPs transitions Idle to Ready.
- Send Run I/O data command, data = 0. DOPs transitions Ready to Run.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 0
- Request Set_Attribute_Single, Idle_State = 0.

6.6) If the Fault_Value attribute is not settable, skip this step of the test.

Fault Value Test

- Request Set_Attribute_Single, Fault_Value = 1.
- Request Set_Attribute_Single, Value = 0.
- Allow the I/O message connection to time out, transition Run to Fault.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 1.
- Reset or Create/Allocate the I/O Connection as done for step 6.1, transition Fault to Ready
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 1.
- Request Set_Attribute_Single, Fault_Value = 0.
- Send Run I/O data command. DOPs transitions Ready to Run.

6.7) If the Fault_State attribute is not settable, skip this step of the test.

Fault State Test

- Request Set_Attribute_Single, Fault_State = 1 (Hold Last State).
- Request Set_Attribute_Single, Value = 1.
- Allow the I/O message connection to time out, transition Run to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.

- Reset or Create/Allocate the I/O Connection as done for step 6.1, transition Fault to Ready
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Send Run I/O data command. DOPs transitions Ready to Run.
- Request Set_Attribute_Single, Fault_State = 0.

6.8) Value after I/O Connection is Deleted

- Request Set_Attribute_Single, Value = 1.
- Release/Close the I/O Connection.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.

6.9) For each instance of the DOG object, if Instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

6.10) For each instance of the DOG object, if instance attribute 4, Binding, is implemented,

- Request a Get_Attribute_Single, Number_of_Bound_Instances.
- Request a Get_Attribute_Single, Binding.
  Pass: Success_Response with a valid DOP Instance Id.
- Request a Get_Attribute_Single, Value, for each DOP instance in the Binding list.
  Pass: Success_Response, value for each DOP instance.

6.10.1) Check Attributes of Bound Instances

- Request a Get_Attribute_Single, Fault_Action, Fault_Value, Idle_Action, Idle_Value, and Run_Idle_Command implemented in the DOG.
- Request a Set_Attribute_Single for each attribute of the bound instance(s).
  Pass: Error_Response 94 0E FF, attribute not settable (or 14, attribute not implemented).
- Request a Get_Attribute_Single, for the attribute, for each DOP instance.
  Pass: Success_Response, point value = group value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.17 Discrete Group Object Test x1F

This section defines the conformance test for the Discrete Group object. This test is required when the Discrete Group object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Discrete Group object.

2) Instance attributes access rules for the Discrete Group object.

3) Common Service implementations for class and instance.

4) Object–specific service implementations for class and instance.

5) Conformance when incorrect data is used to access attributes and services.

6) Object behavior accessible from the network.

6.1) Implementation of the Attributes_List attribute.

6.2) Implementation of the Bindings attribute.

Procedure Definition

- Initialization
  The Discrete Group object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT, Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(5..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 0 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[USINT (1..104), Attribute_Not_Supported]
Attributes_List (02)	[USINT[Number_of_Attributes], Attribute_Not_Supported]
Number_of_Bound_Instances (03)	[USINT]
Binding, Class/Instance List (04)	[(UNIT (x08, x09, x1D, x1E), UINT (1..65535))[Number_of_Bound_Instances], Attribute_Not_Supported]
Status (05)	[BOOL]
Undefined (06..99)	[Attribute_Not_Supported], Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 0 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected values, Responses]
All Attributes (00..99)	[Service_Not_Supported (x08)]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] If no attributes are supported
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	**Note**: If either Number_of_Attributes or Number_of_Bound_Instances = 0, the corresponding list attribute is not returned. [(USINT, USINT[Number_of_Instances], USINT, (UINT (x08, x09, x1D, x1E), UINT (1..65535)){Number_of_Bound_Instances], BOOL), Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
(15..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests

No additional testing required.

6) Behavior Tests

6.1) For each Discrete Group object instance, if instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.

Pass: Success_Response and List size = Number_of_Attributes.

- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

6.2) For each Discrete Group object instance, if instance attribute 4, Binding, is implemented,

- Request a Get_Attribute_Single, Number_of_Bound_Instances.
- Request a Get_Attribute_Single, Binding.
  Pass: Success_Response with a valid Class Id = x08, x09, x1D, x1E and valid Instance Id.
- Request a Get_Attribute_Single, attribute 3, for each instance in the Binding list.
  Pass: Success_Response or Error_Response that is **NOT** 94 16 FF, Object_Does_Not_Exist

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.18 Analog Input Group Object Test x20

This section defines the conformance test for the Analog Input Group (AIG) object. This test is required when the Analog Input Group object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Analog Input Group object.

2) Instance attributes access rules for the Analog Input Group object.

3) Common Service implementations for class and instance.

4) Object–specific service implementations for class and instance.

5) Conformance when incorrect data is used to access attributes and services.

5.1) Response when Set_Attributes_All is requested with invalid data.

6) Object behavior accessible from the network.

6.1) Implementation of the Attributes_List attribute.

6.2) Implementation of the Bindings attribute.

Procedure Definition:

- Initialization
  The Analog Input Group object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT, Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT, Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[USINT (1..108), Attribute_Not_Supported]
Attributes_List (02)	[USINT[Number_of_Attributes], Attribute_Not_Supported]
Number_of_Bound_Instances (03)	[USINT, Attribute_Not_Supported]
Binding, Instance List (04)	[UINT[Number_of_Bound_Instances], Attribute_Not_Supported]
Status (05)	[BOOL (0, 1), Attribute_Not_Supported]
Owner_Vendor (06)	[UINT, Attribute_Not_Supported]
Owner_Serial_No (07)	[(UDINT, Attribute_Not_Supported]
Value_Type (08)	[USINT (0..9, 100), Attribute_Not_Supported]
Reserved (09)	[Attribute_Not_Supported]
Temp_Scale (10)	[BOOL, Attribute_Not_Supported]
Undefined (11..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Num_Attributes (01)	[Service_Not_Supported, Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes_List (02)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Bound_Instances (03)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Binding, Instance List (04)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Status (05)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Owner_Vendor (06)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Owner_Serial_No (07)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Value_Type (08)	[Success_Response, Attribute_Not_Supported]
Reserved (09)	[Service_Not_Supported, Attribute_Not_Supported]
Temp_Scale (10)	[Success_Response, Attribute_Not_Supported]
Undefined (11..99)	[Service_Not_Supported, Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]

Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] If no attributes are supported
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	**Note**: If Number_of_Attributes = 0, list is not returned. [USINT(1..255), USINT[Number_of_Attributes], USINT, UINT[Number_of_Bound_Instances], BOOL, UINT, (UDINT, USINT (0..9, 100), BOOL), Service_Not_Supported]
Set_Attributes_All (02)	[Success_Response, Service_Not_Supported]
(03..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
(15..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests

- If Attributes 8 is settable, Set_Attribute_Single, Attribute = 8, value = 255.

Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.

- If Attributes 10 is settable, Set_Attribute_Single, Attribute = 10, value = 255.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.

5.1) If Set_Attributes_All is implemented for the instance,

- Request a Set_Attributes_All using too much data.
  Pass: Error_Response 94 15 FF, Too_Much_Data.
- Request a Set_Attributes_All using too little data.
  Pass: Error_Response 94 13 FF, Not_Enough_Data.

6) Behavior Tests

6.1) For each AIG object instance, if instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

6.2) For each AIG object instance, if instance attribute 4, Binding, is implemented,

- Request a Get_Attribute_Single, Number_of_Bound_Instances.
- Request a Get_Attribute_Single, Binding.
  Pass: Success_Response with a valid AIP Instance Id.
- Request a Get_Attribute_Single, Value, for each AIP instance in the Binding list.
  Pass: Success_Response, value for each AIP instance.

6.2.1) Check Attributes of Bound Instances

- Request a Get_Attribute_Single, Value_Data_Type implemented in the AIG.
- Request a Set_Attribute_Single for each attribute of the bound instance(s).
  Pass: Error_Response 94 0E FF, attribute not settable (or 14, attribute not implemented).
- Request a Get_Attribute_Single, for the attribute, for each AIP instance.
  Pass: Success_Response, point value = group value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.19 Analog Output Group Object Test x21

This section defines the conformance test for the Analog Output Group (AOG) object. This test is required when the Analog Output Group object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Analog Output Group object.

2) Instance attributes access rules for the Analog Output Group object.

3) Common Service implementations for class and instance.

4) Object–specific and Reserved service implementations for class and instance.

5) Conformance when incorrect data is used to access attributes and services.

5.1) response when Set_Attributes_All is requested with invalid data.

5.2) response when Set_Attribute_Single is requested with invalid data.

6) Object behavior accessible from the network.

6.1) behavior of the Run and Idle commands before I/O connection is established.

6.2) default behavior of the Idle_State and Idle_Value attributes.

6.3) default behavior of the Fault_State and Fault_Value attributes.

6.4) behavior of Idle_Value attribute, if Settable.

6.5) behavior of Idle_State attribute, if Settable.

6.6) behavior of Fault_Value attribute, if Settable.

6.7) behavior of Fault_State attribute, if Settable.

6.8) value after I/O Connection is Deleted

6.9) implementation of the Attributes_List attribute.

6.10) implementation of the Bindings attribute.

Procedure Definition

- Initialization
  The AOG object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(9..199), Service_Not_Supported, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[USINT(1..112), Attribute_Not_Supported]
Attributes_List (02)	[USINT[Number_of_Attributes], Attribute_Not_Supported]
Bound_Instances (03)	[USINT, Attribute_Not_Supported]
Binding, Instance List (04)	[UINT[Bound_Instances], Attribute_Not_Supported]
Status (05)	[BOOL, Attribute_Not_Supported]
Owner_Vendor_Id (06)	[UINT, Attribute_Not_Supported]
Owner_Serial_No (07)	[(UDINT, Attribute_Not_Supported]
Value_Data_Type (08)	[USINT (0..9, 100), Attribute_Not_Supported]
Command (09)	[BOOL]
Fault_State (10)	[BOOL, Attribute_Not_Supported]
Fault_Value (11)	[Type specified by Attribute 8, Attribute_Not_Supported]
Idle_State (12)	[BOOL, Attribute_Not_Supported]
Idle_Value (13)	[Type specified by Attribute 8, Attribute_Not_Supported]
Undefined (14..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes_List (02)	[Attribute_Not_Settable, Attribute_Not_Supported]
Number_of_Bound_Instances (03)	[Attribute_Not_Settable, Attribute_Not_Supported]
Binding, Instance List (04)	[Attribute_Not_Settable, Attribute_Not_Supported]
Status (05)	[Attribute_Not_Settable, Attribute_Not_Supported]

Attribute Name (ID)	[Expected values, Responses]
Owner_Vendor_Id (06)	[Attribute_Not_Settable, Attribute_Not_Supported]
Owner_Serial_No (07)	[Attribute_Not_Settable, Attribute_Not_Supported]
Attribute Name (ID)	[Expected values, Responses]
Value_Data_Type (08)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Command (09)	[Success_Response]
Fault_State (10)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Fault_Value (11)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Idle_State (12)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Idle_Value (13)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Undefined (14..99)	[Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] If no attributes are supported
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	**Note:** If either Number_of_Attributes or Number_of_Bound Instances = 0, the corresponding list attribute is not returned. [(USINT, USINT[Number_of_Attributes], USINT, UINT[Bound_Instances], BOOL, UINT, UDINT, USINT (0..9, 100), BOOL, BOOL, Fault_Value, BOOL, Idle_Value), Service_Not_Supported] **Note:** Data type for Fault_Value and Idle_Value depends on Attribute 8, Value_Data_Type
Set_Attributes_All (02)	[Success_Response, Service_Not_Supported]
(03..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable]
(17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: Expected responses for Object–specific Services addressed to the Class.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: Expected responses for Reserved Services addressed to the Class.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) If Set_Attributes_All is implemented for the instance,

- Request a Set_Attributes_All using too much data.
  Pass: Error_Response 94 15 FF, Too_Much_Data.
- Request a Set_Attributes_All using too little data.
  Pass: Error_Response 94 13 FF, Not_Enough_Data.

5.2)

- Request a Set_Attribute_Single, Value_Data_Type = 144.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.
- Request a Set_Attribute_Single, Command = 254.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.
- Request a Set_Attribute_Single, Fault_State = 254.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.
- Request a Set_Attribute_Single, Idle_State = 254.
  Pass: Error_Response 94 09 FF, Invalid_Attribute_Value.

6) Behavior Test.

Note: This test description was written for both dynamically created and Predefined Master/Slave Connection Set connections. For Predefined Master/Slave Connections, the test uses a Group2 I/O message, depending on the I/O connection, to send Idle and Run commands. The Run I/O messages are the I/O data for the test and cause each AOP in the AOG to transition to Run.

Note: Set the Command attribute as specified to put the AOPs in the Idle or Ready state. Follow the Behavior Test without using the Command attribute. Then, repeat steps 6.1 through 6.7 using the Command Attribute to put the AOP in the Idle or Ready state.

Note: A Get_Attribute_Single request to the Command attribute must always return the value zero regardless of the value that is used with a Set_Attribute_Single request. This check is done for each Get_Attribute_Single request to the Command attribute but is only noted for the first two requests.

Note: Run command I/O messages contain the number of bytes specified by the Consumed Connection Size attribute and use non zero values unless otherwise noted.

6.1) Create/Allocate I/O Connection. If necessary, configure the I/O connection to access grouped AOP value attributes. Set the connection Watchdog_Timeout_Action to transition to timed out.

- Set Command Attribute = 0.
  Pass: Error_Response 94 0C FF, Object_State_Conflict.
- Send Idle I/O data command.
  Pass: No_Response.
- Set Command Attribute = 1.
  Pass: Error_Response 94 0C FF, Object_State_Conflict.
- Send Run I/O data command.
  Pass: No_Response.
- Request Set_Attribute_Single, Value = 1.
- Request Set_Attribute_Single, I/O Connection WatchDog_Timeout_Action = Time Out.

AOPs transitions Available to Ready when connection transitions to Established state.

- Request Set_Attribute_Single, I/O Connection EPR = 500 milliseconds.
- For Dynamic I/O connections, request Apply_Attributes.

6.2) Idle Value Default Test

- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Set Command Attribute = 0. AOPs transitions Ready to Idle.
- Get the Command Attribute
  Pass: Success_Response, Value = 0.
- Send Idle I/O data command.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.
- Set Command Attribute = 1. AOPs transitions Idle to Ready.
- Request Get_Attribute_Single Command attribute.
  Pass: Success_Response, Value = 0.
- Send Run I/O data command, data = 1. AOPs transitions Ready to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Send Run I/O data command, data = 0. AOPs transitions Run to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.
- Request Set_Attribute_Single, Value = 1.

- Set Command Attribute = 0. AOPs transitions Run to Idle.
- Send Idle I/O data command.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.
- Set Command Attribute = 1. AOPs transitions Idle to Ready.
- Send Run I/O data command, data = 1. AOPs transitions Ready to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.

6.3) Fault Value Default Test

- Allow the I/O message connection to time out, transition Run to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.
- Reset or Create/Allocate the I/O Connection as done for step 6.1. transition Fault to Ready
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.
- Request Set_Attribute_Single, Value = 1.
- Allow the I/O connection to time out, transition Ready to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Reset or Create/Allocate the I/O Connection as done for step 6.1. transition Fault to Ready
- Set Command Attribute = 0. AOPs transitions to Idle state.
- Send Idle I/O data command, transition Ready to Idle.
- Request Set_Attribute_Single, Value = 1.
- Allow the I/O connection to time out, transition Idle to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Reset or Create/Allocate the I/O Connection as done for step 6.1. transition Fault to Ready
- Set Command Attribute = 1. AOPs transitions to Ready state.
- Send Run I/O data command, transition Ready to Run.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.

6.4) If Idle_Value attribute is not settable, skip this step of the test.
Idle Value Test

- Request Set_Attribute_Single, Idle_Value = 1.
- Request Set_Attribute_Single, Value = 0.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.
- Set Command Attribute = 0. AOPs transitions Run to Idle.
- Send Idle I/O data command.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Set Command Attribute = 1. AOPs transitions Idle to Ready.

- Send Run I/O data command. AOPs transitions Ready to Run.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 1.
- Request Set_Attribute_Single, Idle_Value = 0.

6.5) If the Idle_State attribute is not settable, skip this step of the test.
Idle State Test

- Request Set_Attribute_Single, Idle_State = 1. (Hold Last Value)
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 1.
- Set Command Attribute = 0. AOPs transitions Run to Idle.
- Send Idle I/O data command. AOPs transitions Run to Idle.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 1.
- Set Command Attribute = 1. AOPs transitions Idle to Ready.
- Send Run I/O data command, data = 0. AOPs transitions Ready to Run.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 0
- Request Set_Attribute_Single, Idle_State = 0.

6.6) If the Fault_Value attribute is not settable, skip this step of the test.
Fault Value Test

- Request Set_Attribute_Single, Fault_Value = 1.
- Request Set_Attribute_Single, Value = 0.
- Allow the I/O message connection to time out, transition Run to Fault.
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 1.
- Reset or Create/Allocate the I/O Connection as done for step 6.1, transition Fault to Ready
- Request Get_Attribute_Single, Value
  Pass: Success_Response, Value = 1.
- Request Set_Attribute_Single, Fault_Value = 0.
- Send Run I/O data command. AOPs transitions Ready to Run.

6.7) If the Fault_State attribute is not settable, skip this step of the test.
Fault State Test

- Request Set_Attribute_Single, Fault_State = 1 (Hold Last State).
- Request Set_Attribute_Single, Value = 1.
- Allow the I/O message connection to time out, transition Run to Fault.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Reset or Create/Allocate the I/O Connection as done for step 6.1, transition Fault to Ready
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 1.
- Send Run I/O data command. AOPs transitions Ready to Run.
- Request Set_Attribute_Single, Fault_State = 0.

6.8) Value after I/O Connection is Deleted

- Request Set_Attribute_Single, Value = 1.
- Release/Close the I/O Connection.
- Request Get_Attribute_Single, Value.
  Pass: Success_Response, Value = 0.

6.9) For each instance of the AOG object, if Instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

6.10) For each AOG object instance, if instance attribute 4, Binding, is implemented,

- Request a Get_Attribute_Single, Number_of_Bound_Instances.
- Request a Get_Attribute_Single, Binding.
  Pass: Success_Response with a valid AOP Instance Id.
- Request a Get_Attribute_Single, Value, for each AOP instance in the Binding list.
  Pass: Success_Response, value for each AOP instance.

6.10.1) Check Attributes of Bound Instances

- Request a Get_Attribute_Single, Value_Data_Type, Run_Idle_Command, Fault_Action, Fault_Value, Idle_Action, and Idle_Value implemented in the AOG.
- Request a Set_Attribute_Single for each attribute of the bound instance(s).
  Pass: Error_Response 94 0E FF, attribute not settable (or 14, attribute not implemented).
- Request a Get_Attribute_Single, for the attribute, for each AOP instance.
  Pass: Success_Response, point value = group value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.20 <u>Analog Group Object Test x22</u>

This section defines the conformance test for the Analog Group object. This test is required when the Analog Group object is implemented in the device.

Functional Description

This test verifies the:

1) class attribute access rules for the Analog Group object.

2) instance attribute access rules for the Analog Group object.

3) Common Service implementations for class and instance.

4) Object–specific service implementations for class and instance.

5) conformance when incorrect data is used to access attributes and services.

5.1) response when Set_Attribute_Single is requested with invalid data.

6) object behavior accessible from the network.

6.1) implementation of the Attributes_List attribute.

6.2) implementation of the Bindings attribute.

Procedure Definition

- Initialization
  The Analog Group object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from

instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[USINT (1..107), Attribute_Not_Supported]
Attributes_List (02)	[USINT[Number_of_Attributes], Attribute_Not_Supported]
Bound_Instances (03)	[USINT (0..255)]
Binding, Class/Instance List (04)	[(UINT (x0A, x0B, x20, x21), UINT (1..65535))[Bound_Instances], Attribute_Not_Supported]
Status (05)	[BOOL (0, 1), Attribute_Not_Supported]
Owner_Vendor (06)	[UINT, Attribute_Not_Supported]
Owner_Serial_No (07)	[(UDINT, Attribute_Not_Supported]
Value_Data_Type (08)	[USINT (0..9, 100), Attribute_Not_Supported]
Undefined (09..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Num_Attributes (01)	[Service_Not_Supported, Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes_List (02)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Bound_Instances (03)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Binding, Class/Instance List (04)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Status (05)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Owner_Vendor (06)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Owner_Serial_No (07)	[Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Value_Data_Type (08)	[Success_Response, Service_Not_Supported, Attribute_Not_Supported]
Undefined (09..99)	[Service_Not_Supported, Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] If no attributes are supported
(15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	**Note**: If either Number_of_Attributes or Number_of_Bound_Instances = 0, the corresponding list attribute is not returned. [(USINT, USINT[Number_of_Attributes], USINT, (UINT(x0A, x0B, x20, x21), UINT(1..65535))[Bound_Instance], BOOL, UINT, UDINT, USINT (0..9, 100)), Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value, Service_Not_Supported]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Service_Not_Supported] [Success_Response] If Attribute 8 is settable
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) For these attributes implemented with Set access,

- Request a Set_Attribute_Single service, Value_Data_Type = 240.
- Request a Set_Attribute_Single service, Fault_State = 248.
- Request a Set_Attribute_Single service, Idle_State = 248.

Pass: For each attribute, Error_Response 94 09 FF, Invalid_Attribute_Value.

6) Behavior Tests.

6.1) For each Analog Group object instance, if instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

6.2) For each Discrete Group object instance, if instance attribute 4, Binding, is implemented,

- Request a Get_Attribute_Single, Number_of_Bound_Instances.
- Request a Get_Attribute_Single, Binding.
  Pass: Success_Response with a valid Class Id = x0A, x0B, x20, x21 and valid Instance Id.
- Request a Get_Attribute_Single, attribute 3, for each instance in the Binding list.
  Pass: Success_Response or Error_Response that is **NOT** 94 16 FF, Object_Does_Not_Exist

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.21 Position Sensor Object Test x23

This section defines the conformance test for the Position Sensor. This test is required when the Position Sensor is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Position Sensor.
2) Instance attributes access rules for the Position Sensor.
3) Common Service implementations for class and instance.
4) Object–specific and Reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
5.1) start list of Error Tests.
6) Object behavior accessible from the network.
6.1) start list of Behavior Tests.

Procedure Definition

- Initialization
  The Position Sensor must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 5000ms.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[SUCCESS UINT (2)]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT (1..199), UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT (1..99), UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(6..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(0..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (02..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from

instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request and refer to CIP Vol 1. Edition 3.6, Table 5-23.3 Position Sensor Object Instance Attributes for details. Note: Instance attrbiute 11 (Position Sensor Type) must be in the range 0..12.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Required Attributes	[Success]
Optional Attributes	[expected values, or Attribute_Not_Supported (x14)]
Undefined (51..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Attribute_1 (01)	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
Attribute_2 (02)	[Success_Response, Service_Not_Supported, Attribute_Not_Settable]
Undefined (03..99)	[Service_Not_Supported, Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] if no attributes are supported
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (code)	[Expected Responses]
(00..13)	[Service_Not_Supported (x08)]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) Error value test for settable attributes

- Set too much data to the attribute

  Pass: The DUT returns 0x15 (Too much data).

- Set less data to the attribute
  Pass: The DUT returns 0x13 (Not enough data).

- Set an invalid value to the attribute
  Pass: The DUT returns 0x09 (Invalid attribute value).

6) Behavior tests

For the details of the Pass / Fail criteria, please follow the CT Test Tool Position Sensor Test itself.

6.1) Attributes and services relationship verification

  - Attribute 3 and 10 shall not be implemented at the same time.
  - Verify conditions for the Conditional attributes

6.2) NV attributes behavior verification

6.3) Preset Value and Offset Value Test

6.4) AutoZero and Zero Offset Tests

6.5) CAM Tests

6.6) Position State Register, Position Low/High Limit Test

6.7) If the test is running in Development Mode, the DUT needs to pass a Position Sensor Interactive Testing, which verifies the sensor's reading according to its moving direction and check its direction toggling behavior.

4.22 Position Controller Supervisor Object Test x24

This section defines the conformance test for the Position Controller Supervisor object. This test is required when the Position Controller Supervisor object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Position Controller Supervisor object.

2) Instance attributes access rules for the Position Controller Supervisor object.

3) Common Service implementations for class and instance.

4) Object–specific and Reserved service implementations for class and instance.

5) Conformance when incorrect data is used to access attributes and services.

5.1) Error responses for set attribute within valid data.

5.2) Error responses for the Command Message Errors

6) Object behavior accessible from the network.

6.1) Responses for the Command Messages

6.2) Implementation of Attribute List

Procedure Definition

- Initialization
  The Position Controller Supervisor object must be available. Log the object name and test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 5000 ms.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT(2)]
Max_Instance (02)	[UINT(1..7), Attribute_Not_Supported]
Num_Instances (03)	[UINT(1..7), Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(33..199), Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(7..199), Attribute_Not_Supported]
Undefined (08..31)	[Attribute_Not_Supported]
Consumed_Command_Msg (32)	[BYTE[8]] defined by profile
Produced_Command_Msg(33)	[BYTE[8]] defined by profile
Undefined (34..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported (x14)]
Any Attribute (01..07)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported (x14)]
Any Attribute (08..31)	[Attribute_Not_Supported]
Consumed_Command_Msg (32)	[Success_Response]
Produced_Response_Msg (33)	[Attribute_Not_Settable]
Any Attribute (34..99)	[Attribute_Not_Supported]

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

Pass: Instance 1 through max_axis are required.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[USINT(4..128), Attribute_Not_Supported]
Attributes_List (02)	[USINT[Number_of_Attributes], Attribute_Not_Supported]
Axis_Number (03)	[USINT(1..7)]
Reserved (04)	[Attribute_Not_Supported]
General_Fault (05)	[BOOL]
Command_Message_Type (06)	[USINT(1..31)]
Response_Message_Type (07)	[USINT(1..31)]
Fault_Input (08)	[BOOL, Attribute_Not_Supported]
Fault_Input_Action (09)	[USINT(0..3, 128..255)], Attribute_Not_Supported]
Home_Action (10)	[USINT(0..4, 128..255)], Attribute_Not_Supported]
Home_Action_Level (11)	[BOOL, Attribute_Not_Supported]
Home_Arm (12)	[BOOL, Attribute_Not_Supported]
Index_Action (13)	[USINT(0..3, 128..255)], Attribute_Not_Supported]
Index_Action_Level (14)	[BOOL, Attribute_Not_Supported]
Index_Arm(15)	[BOOL, Attribute_Not_Supported]
Home_Input_Level(16)	[BOOL, Attribute_Not_Supported]
Home_Position (17)	[DINT, Attribute_Not_Supported]
Index_Position (18)	[DINT, Attribute_Not_Supported]
Registration_Action (19)	[USINT(0..5, 128..255)], Attribute_Not_Supported]
Registration_Active_Level (20)	[BOOL, Attribute_Not_Supported]
Registration_Arm (21)	[BOOL, Attribute_Not_Supported]
Registration_Input_Level(22)	[BOOL, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Registration_Offset (23)	[DINT, Attribute_Not_Supported]
Registration_Position (24)	[DINT, Attribute_Not_Supported]
Follow_Enable (25)	[BOOL, Attribute_Not_Supported]
Follow_Axis (26)	[USINT, Attribute_Not_Supported]
Follow_Divisor (27)	[DINT, Attribute_Not_Supported]
Follow_Multiplier (28)	[DINT, Attribute_Not_Supported]
Undefined (29..98)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[Attribute_Not_Settable (x0E)], Attribute_Not_Supported]
Attributes_List (02)	[Attribute_Not_Settable, Attribute_Not_Supported]
Axis_Number (03)	[Attribute_Not_Settable]
Reserved (04)	[Attribute_Not_Supported]
General_Fault (05)	[Attribute_Not_Settable]
Command_Message_Type (06)	[Success_Response, Attribute_Not_Settable]
Response_Message_Type (07)	[Success_Response, Attribute_Not_Settable]
Fault_Input (08)	[Attribute_Not_Settable, Attribute_Not_Supported]
Fault_Input_Action (09)	[[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Home_Action (10)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Home_Action_Level (11)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Home_Arm (12)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Index_Action (13)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Index_Action_Level (14)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Index_Arm(15)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Home_Input_Level(16)	[Attribute_Not_Settable, Attribute_Not_Supported]
Home_Position (17)	[Attribute_Not_Settable, Attribute_Not_Supported]
Index_Position (18)	[Attribute_Not_Settable, Attribute_Not_Supported]
Registration_Action (19)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Registration_Active_Level (20)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Registration_Arm (21)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Registration_Input_Level(22)	[Attribute_Not_Settable, Attribute_Not_Supported]
Registration_Offset (23)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Registration_Position (24)	[Attribute_Not_Settable, Attribute_Not_Supported]
Follow_Enable (25)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Follow_Axis (26)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Follow_Divisor (27)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Follow_Multiplier (28)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Undefined (29..98)	[Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (01..13)	[Service_Not_Supported (x08)]
Get_Attribute_Single (14)	[requested value, Attribute_Not_Supported (x14)]
Reserved (15)	[Service_Not_Supported (x08)
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable, Atribute_Not_Supported]
Reserved (17..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (code)	[Expected Responses]
(00..13)	[Service_Not_Supported (x08)]
Get_Attribute_Single (14)	[requested value, Attribute_Not_Supported (x14)]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) Test the implemented attributes accessible by Set_Attribute_Single using invalid data.

- attribute 6, Command_Message_Type, value = 255 for this and following attributes
- attribute 7, Response_Message_Type
- attribute 11, Home_Active_Level
- attribute 12, Home_Arm
- attribute 14, Index_Active_Level
- attribute 15, Index_Arm
- attribute 20, Registration_Active_Level
- attribute 21, Registration_Arm
- attribute 25, Follow_Enable
- attribute 9, Fault_Input_Action, value = 6..127 for this and following attributes
- attribute 10, Home_Action
- attribute 13, Index_Action
- attribute 19, Registration_Action

  Pass: Error_Response, 94 09 FF, Invalid_Attribute_Value for all attributes

5.2) I/O Command Message Errors. For Position Controller Profile, use valid command data except for the specified error condition.

- Request a Command Message with less than 8 bytes of data

  Pass: Error Response message x13 with additional code = xff.
- Request a Command Message for each unsupported command type

  Pass: Error Response message x08 with additional code = 01.
- Request a Command Message for invalid response types

  Pass: Error Response message x08 with additional code = 02.
- Request a Command Message for each unsupported command axis

  Pass: Error Response message x05 with additional code = 01.
- Request a Command Message for unsupported response axis

  Pass: Error Response message x05 with additional code = 02.

5.3) I/O Attribute Access Errors.

- Request a Command Message for Get unsupported attribute

  Pass: Error Response message x14 with additional code = x02.
- Request a Command Message for Set unsupported attribute

  Pass: Error Response message x14 with additional code = x01.
- Request a Command Message for Set unsettable attribute

  Pass: Error Response message x0E with additional code = xff.
- Request a Command Message for Set attribute with invalid value

  Pass: Error Response message x09 with additional code = xff.
- Request a Command Message for Get attribute 2 , Attribute List

Pass: Error Response message x11 with additional code = xff, if implemented
Pass: Error Response message x14 with additional code = x02, if not implemented

6) Behavior tests.

6.1) For Position Controller Profile, Command Messages

- Request a Command Message for each supported positioning mode
  Pass: Response message with requested response.

6.1.1) Command Message validation test

- Request a Get Attribute for each supported attribute referenced in the command message
  Pass: Attribute value equals that as sent in the command message.
- Request a Get Attribute, Consumed Command Message
  Pass: Attribute data equals that as sent in the command message.

6.1.2) Response Message validation test

- Request a Get Attribute for each supported attribute referenced in the response message
  Pass: Attribute value equals that as sent in the response message.
- Request a Get Attribute, Produced Response Message
  Pass: Attribute data equals that as sent in the response message.

6.2) Attribute Access, Command Messages

- Request a Command Message for each supported attribute access command
  Pass: Response message with requested response.
- Validate Command and response messages per 6.1.1 and 6.1.2
  Pass: Attribute data equals that as sent in the command/response message.

6.3) If Instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.23 Position Controller Object Test x25

This section defines the conformance test for the Position Controller object. This test is required when the Position Controller object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Position Controller object.

2) Instance attributes access rules for the Position Controller object.

3) Common Service implementations for class and instance.

4) Object–specific and Reserved service implementations for class and instance.

5) Conformance when incorrect data is used to access attributes and services.

5.1) Error responses for set attribute within valid data.

5.2) Error responses for the Command Message Errors

6) Object behavior accessible from the network.

6.1) Responses for the Command Messages

6.2) Implementation of Attribute List

Procedure Definition

- Initialization
  The Position Controller object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT (2)]
Max_Instance (02)	[UINT (1..7), Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..7), Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(1..199), Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(58..199), Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

Pass: Instance 1 through max_axis are required.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.

  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[USINT(7..158)]
Attributes_List (02)	[USINT[Number_of_Attributes]]
Mode (03)	[USINT(0..3), Attribute_Not_Supported]
Position_Units (04)	[DINT(1), Attribute_Not_Supported]
Profile_Units (05)	[DINT(1), Attribute_Not_Supported]
Target_Position (06)	[DINT]
Target_Velocity (07)	[DINT(0..x7FFFFFFF)]
Acceleration (08)	[DINT(0..x7FFFFFFF)]
Deceleration (09)	[DINT(0..x7FFFFFFF), Attribute_Not_Supported]
Incremental_Position_Flag (10)	[BOOL, Attribute_Not_Supported]
Load_Data/Profile_Handshake (11)	[BOOL]
On_Target_Position (12)	[BOOL, Attribute_Not_Supported]
Actual_Position (13)	[DINT, Attribute_Not_Supported]
Actual_Velocity (14)	[DINT(0..x7FFFFFFF), Attribute_Not_Supported]
Commanded_Position(15)	[DINT, Attribute_Not_Supported]
Commanded_Velocity(16)	[DINT(0..x7FFFFFFF), Attribute_Not_Supported]
Enable (17)	[BOOL, Attribute_Not_Supported]
Profile_Type (18)	[USINT(0..2, 128..255)], Attribute_Not_Supported]
Profile _Gain (19)	[DINT, Attribute_Not_Supported]
Smooth_Stop (20)	[BOOL, Attribute_Not_Supported]
Hard_Stop (21)	[BOOL, Attribute_Not_Supported]
Jog_Velocity (22)	[DINT(0..x7FFFFFFF), Attribute_Not_Supported]
Direction (23)	[BOOL, Attribute_Not_Supported]
Reference_Direction (24)	[BOOL, Attribute_Not_Supported]
Torque (25)	[DINT, Attribute_Not_Supported]
Positive_Torque_Limit (26)	[DINT(0..x7FFFFFFF),, Attribute_Not_Supported]
Negative_Torque_Limit (27)	[DINT(x80000001..0),, Attribute_Not_Supported]
Reserved (28)	[Attribute_Not_Supported]
Wrap_Around (29)	[BOOL, Attribute_Not_Supported]
Kp (30)	[INT(0..32767), Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Ki (31)	[INT(0..32767), Attribute_Not_Supported]
Kd (32)	[INT(0..32767), Attribute_Not_Supported]
MaxKi (33)	[INT(0..32767), Attribute_Not_Supported]
KiMode (34)	[USINT(0..1), Attribute_Not_Supported]
Velocity_Feed_Forward (35)	[INT(0..32767), Attribute_Not_Supported]
Accel_Feed_Forward (36)	[INT(0..32767), Attribute_Not_Supported]
Sample_Rate (37)	[INT(0..32767), Attribute_Not_Supported]
Position_Deadband (38)	[USINT, Attribute_Not_Supported]
Feedback_Enable (39)	[BOOL, Attribute_Not_Supported]
Feedback_Resolution (40)	[DINT(0..x7FFFFFFF), Attribute_Not_Supported]
Motor_Resolution (41)	[DINT(0..x7FFFFFFF), Attribute_Not_Supported]
Position_Tracking_Gain (42)	[DINT, Attribute_Not_Supported]
Max_Correction_Velocity (43)	[UINT, Attribute_Not_Supported]
Max_Static_Following_Error (44)	[DINT(0..x7FFFFFFF), Attribute_Not_Supported]
Max_Dynamic_Following_Error (45)	[DINT(0..x7FFFFFFF), Attribute_Not_Supported]
Following_Error_Actoin (46)	[USINT(0..3, 128..255)], Attribute_Not_Supported]
Following_Error_Fault (47)	[BOOL, Attribute_Not_Supported]
Actual_Following_Error (48)	[DINT, Attribute_Not_Supported]
Hard_Limit_Actoin (49)	[USINT(0..2, 128..255)], Attribute_Not_Supported]
Forward_Limit (50)	[BOOL, Attribute_Not_Supported]
Reverse_Limit (51)	[BOOL, Attribute_Not_Supported]
Soft_Limit_Enable (52)	[BOOL, Attribute_Not_Supported]
Soft_Limit_Actoin (53)	[USINT(0..3, 128..255)], Attribute_Not_Supported]
Positive_Soft_Limit_Position (54)	[DINT, Attribute_Not_Supported]
Negative_Soft_Limit_Position (55)	[DINT, Attribute_Not_Supported]
Positive_Limit_Triggered (56)	[BOOL, Attribute_Not_Supported]
Negative_Limit_Triggered (57)	[BOOL, Attribute_Not_Supported]
Load_Data_Complete (58)	[BOOL]
Undefined (59..98)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[Attribute_Not_Settable (x0E)]
Attributes_List (02)	[Attribute_Not_Settable]
Mode (03)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Position_Units (04)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Profile_Units (05)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Target_Position (06)	[Success_Response]
Target_Velocity (07)	[Success_Response]
Acceleration (08)	[Success_Response]
Deceleration (09)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Incremental_Position_Flag (10)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Load_Data/Profile_Handshake (11)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
On_Target_Position (12)	[Attribute_Not_Settable, Attribute_Not_Supported]
Actual_ Position (13)	[Success_Response, Attribute_Not_Supported]
Actual_Velocity (14)	[Attribute_Not_Settable, Attribute_Not_Supported]
Commanded_Position(15)	[Attribute_Not_Settable, Attribute_Not_Supported]
Commanded_Velocity(16)	[Attribute_Not_Settable, Attribute_Not_Supported]
Enable (17)	[Success_Response, Attribute_Not_Supported]
Profile_Type (18)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Profile_Gain (19)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Smooth_Stop (20)	[Success_Response, Attribute_Not_Supported]
Hard_Stop (21)	[Success_Response, Attribute_Not_Supported]
Jog_Velocity (22)	[Success_Response, Attribute_Not_Supported]
Direction (23)	[Success_Response, Attribute_Not_Supported]
Reference_Direction (24)	[Success_Response, Attribute_Not_Supported]
Torque (25)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Positive_Torque_Limit (26)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Negative_Torque_Limit (27)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Reserved (28)	[Attribute_Not_Supported]
Wrap_Around (29)	[Attribute_Not_Settable, Attribute_Not_Supported]
Kp (30)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Ki (31)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Kd (32)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
MaxKi (33)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
KiMode (34)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Velocity_Feed_Forward (35)	[Success_Response, Attribute_Not_Supported]
Accel_Feed_Forward (36)	[Success_Response, Attribute_Not_Supported]
Sample_Rate (37)	[Attribute_Not_Settable, Attribute_Not_Supported]
Position_Deadband (38)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Feedback_Enable (39)	[Success_Response, Attribute_Not_Supported]
Feedback_Resolution (40)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Motor_Resolution (41)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Position_Tracking_Gain (42)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Max_Correction_Velocity (43)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Max_Static_Following_Error (44)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Max_Dynamic_Following_Error (45)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Following_Error_Action (46)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Following_Error_Fault (47)	[Success_Response, Attribute_Not_Supported]
Actual_Following_Error (48)	[Attribute_Not_Settable, Attribute_Not_Supported]
Hard_Limit_Actoin (49)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Forward_Limit (50)	[Attribute_Not_Settable, Attribute_Not_Supported]
Reverse_Limit (51)	[Attribute_Not_Settable, Attribute_Not_Supported]
Soft_Limit_Enable (52)	[Success_Response, Attribute_Not_Supported]
Soft_Limit_Action (53)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Positive_Soft_Limit_Position (54)	[Success_Response, Attribute_Not_Supported]
Negative_Soft_Limit_Position (55)	[Success_Response, Attribute_Not_Supported]
Positive_Limit_Triggered (56)	[Attribute_Not_Settable, Attribute_Not_Supported]
Negative_Limit_Triggered (57)	[Attribute_Not_Settable, Attribute_Not_Supported]
Load_Data_Complete (58)	[Attribute_Not_Settable]
Undefined (59..98)	[Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] if no attributes are supported
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (code)	[Expected Responses]
(00..13)	[Service_Not_Supported (x08)]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.
5.1) Invalid attribute values

- Request a Set_Attribute_Single service with value = -1 for each settable attribute
- attribute 4, Position_Units
- attribute 5, Profile_Units
- attribute 7, Target_Velocity
- attribute 8, Acceleration
- attribute 9, Deceleration
- attribute 22, Jog_Velocity
- attribute 26, Positive_Torque_Limit
- attribute 30, Kp
- attribute 31, Ki
- attribute 32, Kd
- attribute 33, MaxKi
- attribute 35, Velocity_Feed_Forward
- attribute 36, Accel_Feed_Forward
- attribute 40, Feedback_Resolution
- attribute 41, Motor_Resolution
- attribute 43, Max_Correction_Velocity
- attribute 44, Max_Static_Following_Error
- attribute 45, Max_Dynamic_Following_Error
  Pass: Error_Response, 94 09 FF, Invalid Attribute Value.
- Request a Set_Attribute_Single service with value = 0x0AC for each settable attribute
- attribute 10, Incremental_Position_Flag
- attribute 11, Load_Data/Profile_Handshake
- attribute 17, Enable

- attribute 20, Smooth_Stop
- attribute 21, Hard_Stop
- attribute 23, Direction
- attribute 24, Reference_Direction
- attribute 39, Feedback_Enable
- attribute 47, Following_Error_Fault
- attribute 52, Soft_Limit_Enable

  Pass: Error_Response, 94 09 FF, Invalid Attribute Value.
- Request a Set_Attribute_Single service for each settable attribute, value = 0x7F
- attribute 3, Mode
- attribute 18, Profile_Type
- attribute 27, Negative_Torque_Limit
- attribute 34, KiMode
- attribute 46, Following_Error_Action
- attribute 49, Hard_Limit_Action
- attribute 53, Soft_Limit_Action

  Pass: Error_Response, 94 09 FF, Invalid Attribute Value.

5.2) I/O Attribute Access Errors. For Position Controller Profile, use valid command data except for the specified error condition.

- Request a Command Message for Get unsupported attribute

  Pass: Error Response message x14 with additional code = x02.
- Request a Command Message for Set unsupported attribute

  Pass: Error Response message x14 with additional code = x01.
- Request a Command Message for Set unsettable attribute

  Pass: Error Response message x0E with additional code = xff.
- Request a Command Message for Set attribute with invalid value

  Pass: Error Response message x09 with additional code = xff.
- Request a Command Message for Get attribute 2 , Attribute List

  Pass: Error Response message x11 with additional code = xff

6) Behavior tests.

6.1) For Position Controller Profile, Command Messages

- Request a Command Message for each supported attribute access command

  Pass: Response message with requested response.

6.1.1) Command Message validation.

- Request a Get Attribute for each supported attribute referenced in the command message

  Pass: Attribute value equals that as sent in the command message.
- Request a Get Attribute, Position Controller Class attribute Consumed Command Message

  Pass: Attribute data equals that as sent in the command message.

6.1.2) Response Message validation.

- Request a Get Attribute for each supported attribute referenced in the response message

  Pass: Attribute value equals that as sent in the response message.
- Request a Get Attribute, Position Controller Class attribute Produced Response Message

Pass: Attribute data equals that as sent in the response message.

6.2) Instance attribute 2, Attribute_List,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.24 Block Sequencer Object Test x26

This section defines the conformance test for the Block Sequencer object. This test is required when the Block Sequencer object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Block Sequencer object.
2) Instance attributes access rules for the Block Sequencer object.
3) Common Service implementations for class and instance.
4) Object–specific and Reserved service implementations for class and instance.
5.1) Error responses for set attribute within valid data.
5.2) Error responses for the Command Message Errors
6) Object behavior accessible from the network.
6.1) Responses for the Command Messages
6.1) Command Block Programming and Execution

Procedure Definition

- Initialization
  The Block Sequencer object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..255), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..255), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Block (01)	[USINT]
Block Execute (02)	[BOOL]
Current Block (03)	[USINT]
Block Fault (04)	[BOOL]
Block Fault Code (05)	[USINT(0..3), Attribute_Not_Supported]
Counter (06)	[DINT(0..0x7FFFFFFF), Attribute_Not_Supported]
Undefined (07..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Block (01)	[Success_Response]
Block Execute (02)	[Success_Response]
Current Block (03)	[Attribute_Not_Settable (x0E)]]
Block Fault (04)	[Attribute_Not_Settable]
Block Fault Code (05)	[Attribute_Not_Settable, Attribute_Not_Supported]
Counter (06)	[Success_Response , Attribute_Not_Supported]
Undefined (07..99)	[Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] if no attributes are supported
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (code)	[Expected Responses]
(00..13)	[Service_Not_Supported (x08)]
Get_Attribute_Single (14)	[requested value]

Service Name (code)	[Expected Responses]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) Invalid attribute values

- Request a Set_Attribute_Single service for Counter, value = -1
  Pass: Error_Response, 94 09 FF, Invalid Attribute Value.

5.1) I/O Command Message Errors. For Position Controller Profile, use valid command data except for the specified error condition.

- Request a Command Message for Get unsupported attribute
  Pass: Error Response message x14 with additional code = x02.
- Request a Command Message for Set unsupported attribute
  Pass: Error Response message x14 with additional code = x01.
- Request a Command Message for Set unsettable attribute
  Pass: Error Response message x0E with additional code = xff.
- Request a Command Message for Set attribute with invalid value
  Pass: Error Response message x09 with additional code = xff.

6) Behavior tests.

6.1) For Position Controller Profile, Command Messages

- Request a Command Message for each supported attribute access command
  Pass: Response message with requested response.

6.1.1) Command Message validation test.

- Request a Get Attribute for each supported attribute referenced in the command message
  Pass: Attribute value equals that as sent in the command message.
- Request a Get Attribute, Position Controller Class attribute Consumed Command Message
  Pass: Attribute data equals that as sent in the command message.

6.1.2) Response Message validation test.

- Request a Get Attribute for each supported attribute referenced in the response message
  Pass: Attribute value equals that as sent in the response message.
- Request a Get Attribute, Position Controller Class attribute Produced Response Message
  Pass: Attribute data equals that as sent in the response message.

6.2) Block Programming/Execution

Test under development. To be tested in combination with the Command Block Object.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.25 Command Block Object Test x27

This section defines the conformance test for the Command Block object. This test is required when the Command Block object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Command Block object.
2) Instance attributes access rules for the Command Block object.
3) Common Service implementations for class and instance.
4) Object–specific and Reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
5.1) Error responses for the Command Message Errors
6) Object behavior accessible from the network.
6.1) Responses for the Command Messages
6.2) Block Programming and Execution

Procedure Definition

- Initialization
  The Command Block object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..255), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..255), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Block Command (01)	[USINT]
Block Link # (02)	[USINT]
Modify Attribute, Command 01	
Target Class (03)	[USINT]
Target Instance (04)	[USINT]
Attribute # (05)	[USINT]
Attribute Data (06)	[BYTE[]]
Wait Equals, Command 02	
Target Class (03)	[USINT]
Target Instance (04)	[USINT]
Attribute # (05)	[USINT]
Compare Time Out (06)	[DINT(0..0x7FFFFFFF]
Compare Data (07)	[BYTE[]]
Conditional Link > , Command 03	
Target Class (03)	[USINT]
Target Instance (04)	[USINT]
Attribute # (05)	[USINT]
Compare Link (06)	[USINT]
Compare Data (07)	[BYTE[]]
Conditional Link < , Command 04	
Target Class (03)	[USINT]
Target Instance (04)	[USINT]
Attribute # (05)	[USINT]
Compare Link (06)	[USINT]
Compare Data (07)	[BYTE[]]
Delay , Command 06	
Delay (03)	[DINT(1..0x7FFFFFFF]
Trajectory , Command 07	
Target Position (03)	[DINT]
Target Velocity (04)	[DINT]
Incremental (05)	[BOOL]
Trajectory & Wait, Command 08	
Target Position (03)	[DINT]
Target Velocity (04)	[DINT]
Incremental (05)	[BOOL]
Change Velocity, Command 09	

Attribute Name (ID)	**[Expected Values, Responses]**
Target Velocity (03)	[DINT]
Goto Home, Command 10	
Home Offset (03)	[DINT]
Velocity (04)	[DINT]
Goto Index, Command 11	
Index Offset (03)	[DINT]
Velocity (04)	[DINT]
Goto Registration, Command 12	
Registration Offset (03)	[DINT]
Velocity (04)	[DINT]
Undefined (0n..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	**[Expected Responses]**
Undefined (00)	[Attribute_Not_Supported (x14)]
Block Command (01)	[Success_Response]
Block Link # (02)	[Success_Response]
Modify Attribute, Command 01	
Target Class (03)	[Success_Response]
Target Instance (04)	[Success_Response]
Attribute # (05)	[Success_Response]
Attribute Data (06)	[Success_Response]
Wait Equals, Command 02	
Target Class (03)	[Success_Response]
Target Instance (04)	[Success_Response]
Attribute # (05)	[Success_Response]
Compare Time Out (06)	[Success_Response]
Compare Data (07)	[Success_Response]
Conditional Link > , Command 03	
Target Class (03)	[Success_Response]
Target Instance (04)	[Success_Response]
Attribute # (05)	[Success_Response]
Compare Link (06)	[Success_Response]
Compare Data (07)	[Success_Response]
Conditional Link < , Command 04	
Target Class (03)	[Success_Response]
Target Instance (04)	[Success_Response]
Attribute # (05)	[Success_Response]
Compare Link (06)	[Success_Response]

Attribute Name (ID)	[Expected Responses]
Compare Data (07)	[Success_Response]
Delay , Command 06	
Delay (03)	[Success_Response]
Trajectory , Command 07	
Target Position (03)	[Success_Response]
Target Velocity (04)	[Success_Response]
Incremental (05)	[Success_Response]
Trajectory & Wait, Command 08	
Target Position (03)	[Success_Response]
Target Velocity (04)	[Success_Response]
Incremental (05)	[Success_Response]
Change Velocity, Command 09	
Target Velocity (03)	[Success_Response]
Goto Home, Command 10	
Home Offset (03)	[Success_Response]
Velocity (04)	[Success_Response]
Goto Index, Command 11	
Index Offset (03)	[Success_Response]
Velocity (04)	[Success_Response]
Goto Registration, Command 12	
Registration Offset (03)	[Success_Response]
Velocity (04)	[Success_Response]
Undefined (n..99)	[Service_Not_Supported, Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00..13)	[Service_Not_Supported (x08)]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] if no attributes are supported
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (code)	[Expected Responses]
(00..13)	[Service_Not_Supported (x08)]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Supported (x14)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) I/O Command Message Errors. For Position Controller Profile, use valid command data except for the specified error condition.

- Request a Command Message for Get unsupported attribute
  Pass: Error Response message x14 with additional code = x02.
- Request a Command Message for Set unsupported attribute
  Pass: Error Response message x14 with additional code = x01.
- Request a Command Message for Set unsettable attribute
  Pass: Error Response message x0E with additional code = xff.
- Request a Command Message for Set attribute with invalid value
  Pass: Error Response message x09 with additional code = xff.

6) Behavior tests.

6.1) For Position Controller Profile, Command Messages

- Request a Command Message for each supported attribute access command
  Pass: Response message with requested response.

6.1.1) Command Message validation.

- Request a Get Attribute for each supported attribute referenced in the command message
  Pass: Attribute value equals that as sent in the command message.
- Request a Get Attribute, Position Controller Class attribute Consumed Command Message

Pass: Attribute data equals that as sent in the command message.

6.1.2) Response Message validation.

- Request a Get Attribute for each supported attribute referenced in the response message
  Pass: Attribute value equals that as sent in the response message.
- Request a Get Attribute, Position Controller Class attribute Produced Response Message
  Pass: Attribute data equals that as sent in the response message.

6.2) Block Programming and Execution

Test Under Development. To be tested in combination with the Block Sequencer Object.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.26 Motor Data Object Test x28

This section defines the conformance test for the Motor Data object. This test is required when the Motor Data object is implemented in the device.

Functional Description:

This test verifies the:

1) Class attributes access rules for the Motor Data object.
2) Instance attributes access rules for the Motor Data object.
3) Common service implementations for the class and instance.
4) Object-specific and Reserved services for the class and instance.
5) Conformance when incorrect data is used to access attributes and services.

5.1) response when Set_Attribute_Single is requested with invalid data.
5.2) response when Set_Attribute_Single is requested with not enough data.
5.3) response when Set_Attribute_Single is requested with too much data.
5.4) response when Get_Member is requested with invalid data.

6) The Motor Data object behavior accessible from the network.

6.1) Behavior for International Strings.
6.2) Implementation of Attribute List.

Procedure Definition:

- Initialization:
  Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 5000 ms.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT(1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT, Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(6..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(7..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, instance ID zero, to access attributes 00 to 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access Test

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

Note: Instance ID 1 of the Motor Data object must be implemented.

- Request a Get_Attribute_Single service addressed to instance one of the Motor Data class (x28), attributes 00 to 99. Save the return value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
NumAttr (01)	[UINT(1..65535), Attribute_Not_Supported]
Attributes (02)	[USINT[NumAttr], Attribute_Not_Supported]
MotorType (03)	[USINT(0..10)]
CatNumber (04)	[SHORT_STRING, Attribute_Not_Supported]
Manufacturer (05)	[SHORT_STRING, Attribute_Not_Supported]
RatedCurrent (06)	[UINT]
RatedVoltage (07)	[UINT]
RatedPower (08)	[UDINT, Attribute_Not_Supported]
RatedFreq (09)	[UINT, Attribute_Not_Supported]
RatedTemp (10)	[UINT, Attribute_Not_Supported]
MaxSpeed (11)	[UINT, Attribute_Not_Supported]
PoleCount (12)	[UINT, Attribute_Not_Supported]
TorqConstant (13)	[UDINT, Attribute_Not_Supported]
Inertia (14)	[UDINT, Attribute_Not_Supported]
BaseSpeed (15)	[UINT, Attribute_Not_Supported]
RatedFieldCur (16)	[UDINT, Attribute_Not_Supported]
MinFieldCur (17)	[UDINT, Attribute_Not_Supported]
RatedFieldVolt (18)	[UINT, Attribute_Not_Supported]
ServiceFactor (19)	[USINT(1..255), Attribute_Not_Supported]
International_Cat_Number (20)	[STRINGI, Attribute_Not_Supported]
International_Mfg_Name (21)	[STRINGI, Attribute_Not_Supported]
Undefined (22..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to instance ID one, to access attributes 00 to 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Num_Attr (01)	[Attribute_Not_Supported, Attribute_Not_Settable (x0E)]
Attributes (02)	[Attribute_Not_Supported, Attribute_Not_Settable]
MotorType (03)	[Success_Response, Attribute_Not_Settable]

Attribute Name (ID)	[Expected Values, Responses]
CatNumber (04)	[Success_Response, Attribute_Not_Supported]
Manufacturer (05)	[Success_Response, Attribute_Not_Supported]
RatedCurrent (06)	[Success_Response]
RatedVoltage (07)	[Success_Response]
RatedPower (08)	[Success_Response, Attribute_Not_Supported]
RatedFreq (09)	[Success_Response, Attribute_Not_Supported]
RatedTemp (10)	[Success_Response, Attribute_Not_Supported]
MaxSpeed (11)	[Success_Response, Attribute_Not_Supported]
PoleCount (12)	[Success_Response, Attribute_Not_Supported]
TorqConstant (13)	[Success_Response, Attribute_Not_Supported]
Inertia (14)	[Success_Response, Attribute_Not_Supported]
BaseSpeed (15)	[Success_Response, Attribute_Not_Supported]
RatedFieldCur (16)	[Success_Response, Attribute_Not_Supported]
MinFieldCur (17)	[Success_Response, Attribute_Not_Supported]
RatedFieldVolt (18)	[Success_Response, Attribute_Not_Supported]
ServiceFactor (19)	[Success_Response, Attribute_Not_Supported]
International_Cat_Number (20)	[Attribute_Not_Supported, Attribute_Not_Settable]
International_Mfg_Name (21)	[Attribute_Not_Supported, Attribute_Not_Settable]
Undefined (22..99)	[Attribute_Not_Supported]

3) Common Services Test

- Request each Common service 00 - 49 addressed to instance ID zero.
  Pass: The expected response from the table below for each Common service request.

Service Name (Code)	[Expected Response]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value, Service_Not_Supported]
(15..49)	[Service_Not_Supported]

- Request each Common service 00 - 49 addressed to instance ID one.
  Pass: The expected response from the table below for each Common service request.

Service Name (Code)	[Expected Response]
Reserved (00..13)	[Service_Not_Supported (x08)]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Supported (x14), Attribute_Not_Settable]
(17)	[Service_Not_Supported]
Get_Member (18)	[Success_Response, if STRINGI implemented Service_Not_Supported]
Reserved (19..20)	[Service_Not_Supported]
Restore (21)	[Success_Response, Service_Not_Supported, Object_State_Conflict (x0C)]

Service Name (Code)	[Expected Response]
Save (22)	[Success_Response, Service_Not_Supported]
Reserved (23..49)	[Service_Not_Supported]

4) Object-specific and Reserved Services Test

- Request each Object-specific service 75-99 addressed to the class (x28), instance ID zero.
  Pass: The expected response from table below for each Object-specific service request.

Service Name (Code)	[Expected Response]
Object-specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved service 100-255 addressed to the class (x28), instance ID zero.
  Pass: The expected response from table below for each Reserved service request.

Service Name (Code)	[Expected Response]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) Test the implemented attributes accessible by Set_Attribute_Single using invalid data.

- Request a Set_Attribute_Single, instance 1, attribute 3, MotorType = 11.
  Pass: [Invalid_Attribute_Value (x09), Attribute_Not_Settable (x0E)]

5.2) Test the implemented attributes accessible by Set_Attribute_Single using insufficient data.

- Request a Set_Attribute_Single, instance 1. Use xFF for each byte shown for each attribute.
- attribute 3, MotorType with 0 bytes of data.
- attribute 4, CatNumber with 0 bytes of data.
- attribute 5, Manufacturer with 0 bytes of data.
- attribute 6, RatedCurrent, with 1 byte of data.
- attribute 7, RatedVoltage, with 1 byte of data.
- attribute 8, RatedPower, with 2 bytes of data.
- attribute 9, RatedFreq, with 1 byte of data.
- attribute 10, RatedTemp, with 1 byte of data.
- attribute 11, RatedTemp, with 1 byte of data.
- attribute 12, PoleCount, with 1 byte of data.
- attribute 13, TorqConstant, with 2 bytes of data.

- attribute 14, Inertia, with 2 bytes of data.
- attribute 15, BaseSpeed, with 1 byte of data.
- attribute 16, RatedFieldCur, with 2 bytes of data.
- attribute 17, MinFieldCur, with 2 bytes of data.
- attribute 18, RatedFieldVolt, with 1 byte of data.
- attribute 19, ServiceFactor, with 0 bytes of data.
  Pass: Error_Response, 94 13 FF, Not_Enough_Data, for all responses.

5.3) Test the implemented attributes accessible by Set_Attribute_Single using too much data.

- Request a Set_Attribute_Single, instance 1. Use xFF for each byte shown for each attribute.
- attribute 3, MotorType with 2 bytes of data.
- attribute 6, RatedCurrent, with 4 bytes of data.
- attribute 7, RatedVoltage, with 4 bytes of data.
- attribute 8, RatedPower, with 8 bytes of data.
- attribute 9, RatedFreq, with 4 bytes of data.
- attribute 10, RatedTemp, with 4 bytes of data.
- attribute 11, RatedTemp, with 4 bytes of data.
- attribute 12, PoleCount, with 4 bytes of data.
- attribute 13, TorqConstant, with 8 bytes of data.
- attribute 14, Inertia, with 8 bytes of data.
- attribute 15, BaseSpeed, with 4 bytes of data.
- attribute 16, RatedFieldCur, with 8 bytes of data.
- attribute 17, MinFieldCur, with 8 bytes of data.
- attribute 18, RatedFieldVolt, with 4 bytes of data.
  attribute 19, ServiceFactor, with 2 bytes of data.
  Pass: Error_Response 94 15 FF, Too_Much_Data, for all responses.

5.4) Get_Member Test, if implemented.

- Request a Get_Member, Destination_List, MemberId = 255.
  Pass: Error_Response, 94 20 FF, Invalid Parameter

6) Behavior tests.

6.1) If instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

6.2) Check International Strings.

- request a Get_Attribute_Single, Identity, instance 1, Supported_Language_List.
  Pass: Success_Response
- determine the number of identifiers
- request a Get_Member service, member = 0 for each International String attribute
  Pass: Success_Response, valid STRINGI data for active language
- request a Get_Member service, for each member, for each International String attribute
  Pass: Success_Response, valid STRINGI data for requested language

Pass: Error_Response, 94 02 FF, Resource_Unavailable
Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.27 Control Supervisor Object Test x29

This section defines the Control Supervisor object test. This test is required for all AC and DC drives.

Functional Description:

This test verifies the:

1) Class attributes access rules for the Control Supervisor object.
2) Instance attributes access rules for the Control Supervisor object.
3) Common service implementations for the class and instance.
4) Object-specific and Reserved services for the class and instance.
5) Conformance when incorrect data is used to access attributes and services.

5.1) response when Set_Attribute_Single is requested with invalid data.
5.2) response when Set_Attribute_Single is requested with not enough data.
5.3) response when Set_Attribute_Single is requested with too much data.

6) Object behavior accessible from the network.

6.1) behavior of Drive in local control mode.
6.2) behavior of Drive in network control mode.
6.3) behavior of ForceFault/Trip.
6.4) behavior of DN Fault and DN Idle.
6.5) behavior of Non-Volatile attributes
6.6) implementation of Attributes List.

Procedure Definition:

- Initialization

Log the object name and test revision.

- Message Connection

Establish an Explicit Messaging Connection. Set the EPR = 5000ms.

1) Class attribute access rules test

- Request a Get_Attribute_Single service addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT(1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(3..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]

Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance Attribute access rules test

Note: Instance ID 1 of the Control Supervisor object must be implemented.

- Request a Get_Attribute_Single service addressed to instance one of the Control Supervisor class (x29), attributes 00 to 99. Save the return value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
NumAttr (01)	[UINT(1..65535), Attribute_Not_Supported]
Attributes (02)	[USINT[NumAttr], Attribute_Not_Supported]
Run1 (03)	[BOOL]
Run2 (04)	[BOOL, Attribute_Not_Supported]
NetCtrl (05)	[BOOL, Attribute_Not_Supported]
State (06)	[USINT(0..7), Attribute_Not_Supported]
Running1 (07)	[BOOL, Attribute_Not_Supported] **Note:** This attribute is required for drives and servos only
Running2 (08)	[BOOL, Attribute_Not_Supported]
Ready (09)	[BOOL, Attribute_Not_Supported]
Faulted (10)	[BOOL, Attribute_Not_Supported] **Note:** This attribute is required for drives and servos only
Warning (11)	[BOOL, Attribute_Not_Supported]
FaultRst (12)	[BOOL, Attribute_Not_Supported] **Note:** This attribute is required for drives and servos only
FaultCode (13)	[UINT, Attribute_Not_Supported]
WarnCode(14)	[UINT, Attribute_Not_Supported]
CtrlFromNet (15)	[BOOL, Attribute_Not_Supported]
DNFaultMode (16)	[USINT(0,1,2,3,4,100..199), Attribute_Not_Supported]
ForceFault/Trip (17)	[BOOL, Attribute_Not_Supported]
ForceStatus (18)	[BOOL, Attribute_Not_Supported]
Delay (19)	[ITIME, Attribute_Not_Supported]
DNIdleMode (20)	[USINT(0,1,2,3,4,100..199), Attribute_Not_Supported]
ProtectMode (21)	[USINT(0,1,2,3,4,100..199), Attribute_Not_Supported]
CycleCount (22)	[UDINT, Attribute_Not_Supported]
Undefined (23..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
NumAttr (01)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Attributes (02)	[Attribute_Not_Settable, Attribute_Not_Supported]
Run1 (03)	[Success_Response]
Run2 (04)	[Success_Response, Attribute_Not_Supported]
NetCtrl (05)	[Success_Response, Attribute_Not_Supported]
State (06)	[Attribute_Not_Settable, Attribute_Not_Supported]
Running1 (07)	[Attribute_Not_Settable, Attribute_Not_Supported] **Note:** Attribute_Not_Settable for drives and servos.
Running2 (08)	[Attribute_Not_Settable, Attribute_Not_Supported]
Ready (09)	[Attribute_Not_Settable, Attribute_Not_Supported]
Faulted (10)	[Attribute_Not_Settable, Attribute_Not_Supported] **Note:** Attribute_Not_Settable for drives and servos.
Warning (11)	[Attribute_Not_Settable, Attribute_Not_Supported]
FaultRst (12)	[Success_Response, Attribute_Not_Supported] **Note:** Success_Response for drives and servos.
FaultCode (13)	[Attribute_Not_Settable, Attribute_Not_Supported]
WarnCode(14)	[Attribute_Not_Settable, Attribute_Not_Supported]
CtrlFromNet (15)	[Attribute_Not_Settable, Attribute_Not_Supported]
DNFaultMode (16)	[Success_Response, Attribute_Not_Supported]
ForceFault/Trip (17)	[Success_Response, Attribute_Not_Supported]
ForceStatus (18)	[Attribute_Not_Settable, Attribute_Not_Supported]
Delay (19)	[Success_Response, Attribute_Not_Supported]
DNIdleMode (20)	[Success_Response, Attribute_Not_Supported]
ProtectMode (21)	[Success_Response, Attribute_Not_Supported]
CycleCount (22)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Undefined (23..99)	[Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Response]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value, Service_Not_Supported]
(15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (Code)	[Expected Response]
Reserved (00)	[Service_Not_Supported (x08)]
Reserved (01..04)	[Service_Not_Supported]
Reset (05)	[Success_Response]

Service Name (Code)	[Expected Response]
Reserved (06..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Supported, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) Test the implemented attributes accessible by Set_Attribute_Single using invalid data.

- Request a Set_Attribute_Single, instance 1,
- attribute 3, Run1 = F0.
- attribute 4, Run2 = F0.
- attribute 5, NetCtrl = F0.
- attribute 12, FaultRst = F0.
- attribute 16, DNFaultMode = F0.
- attribute 17, ForceFault/Trip = F0.
  Pass: Error_Response, 94 09 FF, Invalid_Attribute_Value for all responses

- attribute 16, DNFaultMode = 5..99, 200..255.
- attribute 20, DNIdleMode = = 5..99, 200..255.
- attribute 20, ProtectMode = = 5..99, 200..255.

Pass: Error_Response, 94 09 FF, Invalid_Attribute_Value for all responses

5.2) Test the implemented attributes accessible by Set_Attribute_Single using insufficient data.

- Request a Set_Attribute_Single, instance 1. Use xF0 for each byte shown for each attribute.
- attribute 3, Run1 with 0 bytes of data.
- attribute 4, Run2 with 0 bytes of data.
- attribute 5, NetCtrl with 0 bytes of data.
- attribute 12, FaultRst with 0 bytes of data.
- attribute 16, DNFaultMode with 0 bytes of data.
- attribute 17, ForceFault/Trip with 0 bytes of data.
  Pass: Error_Response, 94 13 FF, Not_Enough_Data for all responses

5.3) Test the implemented attributes accessible by Set_Attribute_Single using too much data.

- Request a Set_Attribute_Single, instance 1. Use xF0 for each byte shown for each attribute.
- attribute 3, Run1 with 2 bytes of data.
- attribute 4, Run2 with 2 bytes of data.
- attribute 5, NetCtrl with 2 bytes of data.
- attribute 12, FaultRst with 2 bytes of data.
- attribute 16, DNFaultMode with 2 bytes of data.
- attribute 17, ForceFault/Trip with 2 bytes of data.
  Pass: Error_Response, 94 15 FF, Too_Much_Data for all responses

6) Behavior Test
6.1) Network control test part 1. Drive in local control mode. If NetCtrl (attribute 5) is not settable or not implemented, or if CtrlFromNet (attribute 15) is not implemented, or if state (attribute 6) is not implemented, skip step 6.1. Only implemented attributes are tested.

- Send a Set_Attribute_Single service to instance one attribute 3, Run1 = 0.
- Send a Set_Attribute_Single service to instance one attribute 4, Run2 = 0.
- Send a Set_Attribute_Single service to instance one attribute 5, NetCtrl = 0.

Pass: Success_Response

- If a Success_Response was received in the last step, send a Get_Attribute_Single service to instance one attribute 15, CtrlFromNet.
  Pass: CtrlFromNet = 0.

  Note: If the request for local control is refused (return = 1), skip the rest of step 6.1.
- Send a Get_Attribute_Single service to instance one attribute 6, State.
  Note: If State does not = 3, Ready, skip the rest of step 6.1.
- Send a Set_Attribute_Single service to instance one attribute 3, Run1 = 1;
- Send a Get_Attribute_Single service to instance one attribute 6, State.
  Pass: State = 3. Control Supervisor ignores run command and remains in Ready State.
- Send a Get_Attribute_Single service to instance one attribute 7, Running1.
  Pass: Running1 = 0. Control Supervisor ignores run command.
- Send a Get_Attribute_Single service to instance one attribute 9, Ready.
  Pass: Ready = 1.
- Send a Set_Attribute_Single service to instance one attribute 3, Run1 = 0.
- Send a Set_Attribute_Single service to instance one attribute 4, Run2 = 1.

Note: If instance 1 attribute 4 (Run2) is not implemented skip the rest of step 6.1

- Send a Get_Attribute_Single service to instance one attribute 6, State.
  Pass: State = 3. Control Supervisor ignores run command and remains in Ready State.
- Send a Get_Attribute_Single service to instance one attribute 8, Running2.
  Pass: Running2 = 0. Control Supervisor ignores run command.
- Send a Get_Attribute_Single service to instance one attribute 9, Ready.
  Pass: Ready = 1

6.2) Network control test part 2. Drive in network control mode. If NetCtrl (attribute 5) is not settable or not implemented, or if CtrlFromNet (attribute 15) is not implemented, or if state (attribute 6) is not implemented, skip step 6.2. Only implemented attributes are tested.

- Send a Set_Attribute_Single service to instance one attribute 3, Run1 = 0.
- Send a Set_Attribute_Single service to instance one attribute 4, Run2 = 0.
- Send a Set_Attribute_Single service to instance one attribute 5, NetCtrl = 1.
  Pass: Success_Response
- Send a Get_Attribute_Single service to instance one attribute 15, CtrlFromNet.
  Pass: CtrlFromNet = 1

  Note: If the request for network control is refused (return = 0), skip the rest of step 6.2.
- Send a Get_Attribute_Single service to instance one attribute 6, State.
  Note: If the returned value is not 3 = Ready, skip the rest of step 6.2.
- Send a Set_Attribute_Single service to instance one attribute 3, Run1 = 1;
- Send a Get_Attribute_Single service to instance one attribute 6, State.
  Pass: State = 4. Control Supervisor transitions to Enabled.

  Note: The transition time is undetermined. If State=3 (Ready), poll attribute 6 for 10 seconds or until State = 3, Enabled. If State does not transition to Enabled, the test fails.
- Send a Get_Attribute_Single service to instance one attribute 7, Running1.
  Pass: Running1 = 1.
- Send a Get_Attribute_Single service to instance one attribute 9, Ready.
  Pass: Ready = 1.
- Send a Set_Attribute_Single service to instance one attribute 3, Run1 = 0.
- Start a test timer for 10 seconds, when timer expires go to next step.
- Send a Get_Attribute_Single service to instance one attribute 6, State.
  Pass: State = 3.

  Note: Control Supervisor transitions to Ready. This may occur after the drive transitions to Stopping. If State=5 (Stopping), poll attribute 6 until State = 3, Ready. Polling should continue for 10 seconds. If State does not transition to Ready, the test fails.
- Send a Set_Attribute_Single service to instance one attribute 4, Run2 = 1.

Note: If instance 1 attribute 8 (Run2) is not implemented, skip the rest of step 6.2.

- Send a Get_Attribute_Single service to instance one attribute 6, State.
  Pass: State = 4.

  Note: The transition time is undetermined. If State=3 (Ready), poll attribute 6 for 10 seconds or until State = 4, Enabled. If State does not transition to Enabled, the test fails.
- Send a Get_Attribute_Single service to instance one attribute 8, Running2.
  Pass: Running2 = 1.
- Send a Get_Attribute_Single service to instance one attribute 9, Ready.

Pass: Ready = 1.

6.3) Fault/Trip test. If ForceFault/Trip (attribute 17) is not settable, or if state (attribute 6) is not implemented skip step 6.3.

- Send a Set_Attribute_Single service to instance one attribute 17, ForceFault/Trip = 0.
- Send a Set_Attribute_Single service to instance one attribute 17, ForceFault/Trip = 1.
- Send a Get_Attribute_Single service to instance one attribute 6, State.
  Pass: State = 6 or 7.

  Note: Control Supervisor should transition to the Fault_Stop or Faulted state. Continue monitoring the Sate attribute until State = 7, Faulted is returned. Polling should continue for 10 seconds. If State does not transition to Faulted, the test fails.
- Send a Set_Attribute_Single service to instance one attribute 12, FaultRst = 0.
- Send a Set_Attribute_Single service to instance one attribute 12, FaultRst = 1.
  Note: This insures a rising edge of the FaultRst attribute value.
- Send a Get_Attribute_Single service to instance one attribute 6, State.
  Pass: State = 2 or 3

  Note: Control Supervisor should transition to Not_Ready (2) or Ready (3) state. Polling should continue for 10 seconds. If State does not transition in this amount of time, the test fails.

6.4) DNFaultMode DNIdleMode (default)

- Send a Get_Attribute_Single service to instance one attribute 19, Delay, save value.
- Send a Set_Attribute_Single service to instance one attribute 16, DNFaultMode = 0, 1, 3, 4.
- Send a Set_Attribute_Single service to instance one attribute 20, DNIdleMode = 0, 1, 3, 4.
- Open I/O connection, set EPR = 3 seconds.
- Send I/O Idle message.
- Set a test timer for the Delay time, when timer expires, .
- Send a Get_Attribute_Single service to instance one attribute 6, State.
  Pass: State = 6, Fault_Stop.
- Send a Get_Attribute_Single service to instance one attribute 7, Running1.
  Pass: The expected response from table below.

DNFaultMode	0	1	3	4
Running1	0	1	1	1

- Send a Get_Attribute_Single service to instance one attribute 8, Running2.
  Pass: The expected response from table above for Running1.
- Send I/O Run message.
- Send a Get_Attribute_Single service to instance one attribute 6, State.
  Pass: State = 3, Ready.
- Send a Get_Attribute_Single service to instance one attribute 7, Running1.
  Pass: Running1 = 1.
- Send a Get_Attribute_Single service to instance one attribute 8, Running2.
  Pass: Running2 = 1.
- Set a test timer for 5 seconds plus the Delay time, when timer expires, .
- Send a Get_Attribute_Single service to instance one attribute 6, State.

Pass: State = 6, Fault_Stop.

- Send a Get_Attribute_Single service to instance one attribute 7, Running1.
  Pass: The expected response from table below.

DNFaultMode	0	1	3	4
Running1	0	1	1	1

- Send a Get_Attribute_Single service to instance one attribute 8, Running2.
  Pass: The expected response from table above for Running1.
- If DNFaultMode and/or DNIdleMode are settable, repeat step 6.4 with modes = 1, 3, 4.

6.5) Non-Volatile attributes test,

- Request a Set_Attribute_Single for each settable attribute, non-default value.
- Request a Get_Attribute_Single for each implemented attribute.
- Request an Identity Object Reset, type = 0.
- When Network Access is complete, request Get_Attribute_Single for each implemented non-volatile attribute.
  Pass: Each value = pre-Reset value.

6.6) If instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.28 AC/DC Drive Object Test x2A

This section defines the AC/DC Drive object test. This test is required for all AC and DC drives.

Functional Description:

This test verifies the:

1) Class attributes access rules for the AC/DC Drive object.
2) Instance attributes access rules for the AC/DC Drive object.
3) Common service implementations for the class and instance.
4) Object-specific and Reserved services for the class and instance.
5) Conformance when incorrect data is used to access attributes and services.

5.1) response when Set_Attribute_Single is requested with invalid data.
5.2) response when Set_Attribute_Single is requested with not enough data.
5.3) response when Set_Attribute_Single is requested with too much data.

6) Object behavior accessible from the network.

Procedure Definition:

- Initialization
  Log the object name and object and object test software revision and version identifier.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 5000ms.

1) Class attribute access rules test

- Request a Get_Attribute_Single service addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT(1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT, Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(1..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(8..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
NumAttr(01)	[UINT(1..65535), Attribute_Not_Supported]
Attributes (02)	[USINT[NumbAttr], Attribute_Not_Supported]
AtReference (03)	[BOOL, Attribute_Not_Supported]
NetRef (04)	[BOOL]
NetProc (05)	[BOOL, Attribute_Not_Supported (x14)]
DriveMode (06)	[USINT(0..5)]
SpeedActual (07)	[INT]
SpeedRef (08)	[INT]
CurrentActual (09)	[INT, Attribute_Not_Supported]
CurrentLimit (10)	[INT, Attribute_Not_Supported]
TorqueActual (11)	[INT, Attribute_Not_Supported]
TorqueRef (12)	[INT, Attribute_Not_Supported]
ProcessActual (13)	[INT, Attribute_Not_Supported]
ProcessRef (14)	[INT, Attribute_Not_Supported]
PowerActual (15)	[INT, Attribute_Not_Supported]
InputVoltage (16)	[INT, Attribute_Not_Supported]
OutputVoltage (17)	[INT, Attribute_Not_Supported]
AccelTime (18)	[UINT, Attribute_Not_Supported]
DecelTime (19)	[UINT, Attribute_Not_Supported]
LowSpdLimit (20)	[UINT, Attribute_Not_Supported]
HighSpdLimit (21)	[UINT, Attribute_Not_Supported]
SpeedScale (22)	[SINT, Attribute_Not_Supported]
CurrentScale (23)	[SINT, Attribute_Not_Supported]
TorqueScale (24)	[SINT, Attribute_Not_Supported]
ProcessScale (25)	[SINT, Attribute_Not_Supported]
PowerScale (26)	[SINT, Attribute_Not_Supported]
VoltageScale (27)	[SINT, Attribute_Not_Supported]
TimeScale (28)	[SINT, Attribute_Not_Supported]
RefFromNet (29)	[BOOL, Attribute_Not_Supported]
ProcFromNet (30)	[BOOL, Attribute_Not_Supported]
FieldIorV (31)	[BOOL, Attribute_Not_Supported]
FieldVoltRatio (32)	[UINT, Attribute_Not_Supported]
FieldCurSetPt (33)	[UINT, Attribute_Not_Supported]
FieldWkEnable (34)	[BOOL, Attribute_Not_Supported]
FieldFurActual (35)	[INT, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
FieldMinCur (36)	[INT, Attribute_Not_Supported]
Process_Data_Uints (37)	[ENGUNITS, Attribute_Not_Supported]
Speed_Control (38)	[BYTE, Attribute_Not_Supported]
Speed_Status (39)	[BYTE, Attribute_Not_Supported]
Speed_Trip_Time (40)	[UINT, Attribute_Not_Supported]
Max_Rated_Speed (41)	[INT, Attribute_Not_Supported]
Max_Rated_Speed_scale (42)	[SINT, Attribute_Not_Supported]
SpeedStandby (43)	[INT, Attribute_Not_Supported]
SpeedActualDataUnits (44)	[ENGUNITS(0x1f0f), Attribute_Not_Supported]
SpeedRefDataUnits (45)	[ENGUNITS(0x1f0f), Attribute_Not_Supported]
DriveOnHours (46)	[DINT, Attribute_Not_Supported]
Undefined (47..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
NumAttr (01)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes (02)	[Attribute_Not_Settable, Attribute_Not_Supported]
AtReference (03)	[Attribute_Not_Settable, Attribute_Not_Supported]
NetRef (04)	[Success_Response]
NetProc (05)	[Success_Response, Attribute_Not_Supported]
DriveMode (06)	[Success_Response, Attribute_Not_Settable]
SpeedActual (07)	[Attribute_Not_Settable]
SpeedRef (08)	[Success_Response]
CurrentActual (09)	[Attribute_Not_Settable, Attribute_Not_Supported]
CurrentLimit (10)	[Success_Response, Attribute_Not_Supported]
TorqueActual (11)	[Attribute_Not_Settable, Attribute_Not_Supported]
TorqueRef (12)	[Success_Response, Attribute_Not_Supported]
ProcessActual (13)	[Attribute_Not_Settable, Attribute_Not_Supported]
ProcessRef (14)	[Success_Response, Attribute_Not_Supported]
PowerActual (15)	[Attribute_Not_Settable, Attribute_Not_Supported]
InputVoltage (16)	[Attribute_Not_Settable, Attribute_Not_Supported]
OutputVoltage (17)	[Attribute_Not_Settable, Attribute_Not_Supported]
AccelTime (18)	[Success_Response, Attribute_Not_Supported]
DecelTime (19)	[Success_Response, Attribute_Not_Supported]
LowSpdLimit (20)	[Success_Response, Attribute_Not_Supported]
HighSpdLimit (21)	[Success_Response, Attribute_Not_Supported]
SpeedScale (22)	[Success_Response, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
CurrentScale (23)	[Success_Response, Attribute_Not_Supported]
TorqueScale (24)	[Success_Response, Attribute_Not_Supported]
ProcessScale (25)	[Success_Response, Attribute_Not_Supported]
PowerScale (26)	[Success_Response, Attribute_Not_Supported]
VoltageScale (27)	[Success_Response, Attribute_Not_Supported]
TimeScale (28)	[Success_Response, Attribute_Not_Supported]
RefFromNet (29)	[Attribute_Not_Settable, Attribute_Not_Supported]
ProcFromNet (30)	[Attribute_Not_Settable, Attribute_Not_Supported]
FieldIorV (31)	[Success_Response, Attribute_Not_Supported]
FieldVoltRatio (32)	[Success_Response, Attribute_Not_Supported]
FieldCurSetPt (33)	[Success_Response, Attribute_Not_Supported]
FieldWkEnable (34)	[Success_Response, Attribute_Not_Supported]
FieldFurActual (35)	[Attribute_Not_Settable, Attribute_Not_Supported]
FieldMinCur (36)	[Success_Response, Attribute_Not_Supported]
Process_Data_Uints (37)	[Success_Response, Attribute_Not_Supported]
Speed_Control (38)	[Success_Response, Attribute_Not_Supported]
Speed_Status (39)	[Attribute_Not_Settable, Attribute_Not_Supported]
Speed_Trip_Time (40)	[Success_Response, Attribute_Not_Supported]
Max_Rated_Speed (41)	[Attribute_Not_Settable, Attribute_Not_Supported]
Max_Rated_Speed_scale (42)	[Success_Response, Attribute_Not_Supported]
SpeedStandby (43)	[Success_Response, Attribute_Not_Supported]
SpeedActualDataUnits (44)	[Success_Response, Attribute_Not_Supported]
SpeedRefDataUnits (45)	[Success_Response, Attribute_Not_Supported]
DriveOnHours (46)	[Attribute_Not_Settable, Attribute_Not_Supported]
Undefined (47..99)	[Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Response]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value, Service_Not_Supported]
(15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (Code)	[Expected Response]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]

Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Supported, Attribute_Not_Settable (x0E)]
Reserved (17..20)	[Service_Not_Supported]
Restore (21)	[Success_Response, Service_Not_Supported, Object_State_Conflict (x0C)]
Save (22)	[Success_Response, Service_Not_Supported]
Reserved (23..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) Test the implemented attributes accessible by Set_Attribute_Single using invalid data.

- attribute 4, NetRef = 2.
- attribute 5, NetProc = 2.
- attribute 6, DriveMode = xFF.
- attribute 34, FieldWkEnable = 2.
  Pass: Error_Response, 94 09 FF, Invalid_Attribute_Value

5.2) Test the implemented attributes accessible by Set_Attribute_Single using insufficient data.

- Request a Set_Attribute_Single, instance 1. Use xFF for each byte shown for each attribute.
- attribute 4, NetRef with 0 bytes of data.
- attribute 5, NetProc with 0 bytes of data.

- attribute 6, DriveMode with 0 bytes of data.
- attribute 8, SpeedRef with 1 byte of data.
- attribute 10, CurrentLimit with 1 byte of data.
- attribute 12, TorqueRef with 1 byte of data.
- attribute 14, ProcessRef with 1 byte of data.
- attribute 18, AccelTime with 1 byte of data.
- attribute 19, DecelTime with 1 byte of data.
- attribute 20, LowSpeedLimit with 1 byte of data.
- attribute 21, HighSpeedLimit with 1 byte of data.
- attribute 22, SpeedScale with 0 bytes of data.
- attribute 23, CurrentScale with 0 bytes of data.
- attribute 24, TorqueScale with 0 bytes of data.
- attribute 25, ProcessScale with 0 bytes of data.
- attribute 26, PowerScale with 0 bytes of data.
- attribute 27, VoltageScale with 0 bytes of data.
- attribute 28, TimeScale with 0 bytes of data.
- attribute 31, FieldVorI with 0 bytes of data.
- attribute 32, FieldVoltRatio with 1 byte of data.
- attribute 33, FieldCurSetPt with 1 byte of data.
- attribute 34, FieldWkEnable with 0 bytes of data.
- attribute 36, FieldMinCur with 1 byte of data.
  Pass: Error_Response, 94 13 FF, Not_Enough_Data for all responses

5.3) Test the implemented attributes accessible by Set_Attribute_Single using too much data.

- Request a Set_Attribute_Single, instance 1. Use xFF for each byte shown for each attribute.
- attribute 4, NetRef with 2 bytes of data.
- attribute 5, NetProc with 2 bytes of data.
- attribute 6, DriveMode with 2 bytes of data.
- attribute 8, SpeedRef with 4 byte of data.
- attribute 10, CurrentLimit with 4 byte of data.
- attribute 12, TorqueRef with 4 byte of data.
- attribute 14, ProcessRef with 4 byte of data.
- attribute 18, AccelTime with 4 byte of data.
- attribute 19, DecelTime with 4 byte of data.
- attribute 20, LowSpeedLimit with 4 byte of data.
- attribute 21, HighSpeedLimit with 4 byte of data.
- attribute 22, SpeedScale with 2 bytes of data.
- attribute 23, CurrentScale with 2 bytes of data.
- attribute 24, TorqueScale with 2 bytes of data.
- attribute 25, ProcessScale with 2 bytes of data.
- attribute 26, PowerScale with 2 bytes of data.
- attribute 27, VoltageScale with 2 bytes of data.
- attribute 28, TimeScale with 2 bytes of data.
- attribute 31, FieldVorI with 2 bytes of data.
- attribute 32, FieldVoltRatio with 4 byte of data.

- attribute 33, FieldCurSetPt with 4 byte of data.
- attribute 34, FieldWkEnable with 2 bytes of data.
- attribute 36, FieldMinCur with 4 byte of data.
  Pass: Error_Response, 94 15 FF, Too_Much_Data for all responses

6) Behavior tests.

None. The AC/DC Drive object includes attributes that generally affect behavior associated with the physical system the drive is connected to. This type of behavior is not observable from the network.

6.1) If instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.29 <u>Acknowledge Handler Object Test x2B</u>

This section defines the conformance test for the Acknowledge Handler object. This test is required when the Acknowledge Handler object is implemented in the device.

Note: The Acknowledge Handler object test also verifies the behavior of Change of State and Cyclic Connections when producing I/O data and consuming Acknowledgment messages.

Note: This test does not verify the behavior when Acknowledgment messages contain data. Tests for the Ack_With_Data_Path_List and Ack_With_Data_Path attributes will be added in a future revision of this test.

Functional Description

This test verifies the:

1) Class attributes access rules for the Acknowledge Handler object.

2) Instance attributes access rules for the Acknowledge Handler object.

3) Common Service implementations for class and instance.

4) Object–specific and reserved service implementations for class and instance.

5.1) response when Ack_Timer is set with invalid data.

5.2) response when COS_Producing_Connection_Instance is set with invalid data.

5.3) response when Add_AckData_Path is used with invalid data.

Note: Steps 6.1 through 6.15 are done for both Change of State and Cyclic connections.

6.1) the default Acknowledge handler attribute configuration.

6.2) response for setting the Ack_Timer attribute.

6.3) response when COS_Producing_Connection_Instance is set in Inactive state.

6.4) initial I/O data production.

6.5) response when COS_Producing_Connection_Instance is set in Active state.

6.6) I/O production on Transmission Trigger Expiration.

6.7) response when Ack message contains correct amount of data.

6.8) response when Ack message contains incorrect amount of data.

6.9) the Acknowledge handler attributes in Active state.

6.10) behavior of the Retry_Limit attribute.

6.11) I/O connection Inactivity Timer expiration.

6.12) the Acknowledge Handler attributes when I/O connection is in the TimedOut state.

6.13) I/O Connection behavior after Connection TimeOut and Reset service.

6.14) Predefined M/S I/O Connection behavior after Connection TimeOut and Re-Allocate.

6.15) Acknowledge Handler attributes after I/O connection deleted.

6.16) behavior of Dynamic Acknowledge Handler.

Procedure Definition

- Initialization
  The Acknowledge Handler object must be available. Log object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT(1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT, Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(6..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(3..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.
Pass: The Acknowledge Handler object must have instance 1 implemented

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Acknowledge_Timer (01)	[UINT(1..65535)]
Retry_Limit (02)	[USINT(0..255)]
COS Producing Connection_Instance(03)	[UINT(1..65535)] Static [UINT] Dynamic
Ack_List_Size (04)	[(USINT, UINT[size]), Attribute_Not_Supported]
Ack_List (05)	[USINT, Attribute_Not_Supported]
Data_With_Ack_Path_List_Size(06)	[USINT, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Data_With_Ack_Path_List (07)	[USINT, [UINT(1..65535), USINT, EPATH [Path_Size]], Attribute_Not_Supported] see Note 1
Undefined (08..99)	[Attribute_Not_Supported]

Note 1: EPATH encoding is defined in the CIP Specification, Volume 1, Appendix C, Abstract Syntax Encoding for Segment Types. The expected values for a EPATH are as follows.

Segment Type (bits 7, 6, 5) - 1, Logical Segment.
Logical Format (bits 4, 3, 2) - 0, 1, 4; Class ID, Instance ID, Attribute ID.
Logical Data Format (bits 1, 0) - 0, 1; 0 = USINT(1..255), 1 = UINT(1..65535).
An EPATH must specify Class ID, and Instance ID. Attribute ID is optional.

Symbolic and Data Segment Types are unused.
Structure Elements and Connections Point Logical Types are unused.

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Acknowledge_Timer (01)	[Success_Response]
Retry Limit (02)	[Success_Response, Attribute_Not_Settable (x0E)]
COS Producing Connection_Instance (03)	[Attribute_Not_Settable] Static [Success_Response, Object_State_Conflict (x0C)] Dynamic
Ack_List_Size (04)	[Attribute_Not_Settable, Attribute_Not_Supported]
Ack_List (05)	[Attribute_Not_Settable, Attribute_Not_Supported]
Data_With_Ack_Path_List_Size(06)	[Attribute_Not_Settable, Attribute_Not_Supported]
Data_With_Ack_Path_List (07)	[Attribute_Not_Settable, Attribute_Not_Supported]
Undefined (08..99)	[Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
(00..07)	[Service_Not_Supported (x08)]
Create (08)	[Success_Response, Service_Not_Supported]
Delete (09)	[Success_Response, Service_Not_Supported]
(10..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] if no attributes are supported
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (code)	[Expected Responses]

(00..08)	[Service_Not_Supported (x08)]
Delete (09)	[Service_Success, Service_Not_Supported]
(10..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Add_Ack_Data_Path (75)	[Success_Response, Service_Not_Supported (x08)]
Remove_Ack_Data_Path (76)	[Success_Response, Service_Not_Supported (x08)]
Object–specific Codes (77..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) Ack Timer Error Test

- If the Ack_Timer attribute is settable, send a Set_Attribute_Single request, data = 0.
  Pass: Error_Response, 94 09 FF, Invalid_Attribute_Value.

5.1) COS_Producing_Connection_Instance Error Test

- If the COS_Producing_Connection_Instance is settable, send a Set_Attribute_Single request, data = 0.
  Pass: Error_Response, 94 09 FF, Invalid_Attribute_Value.

5.3) Add_AckData_Path Error Test, if Add_AckData_Path is implemented

5.3.1) Add path, invalid path sizes

- Send an Add_AckData_Path request with invalid path size too large.
  Pass: Error_Response, 94 20 FF, Invalid_Parameter.
- Send an Add_AckData_Path request with invalid path size too small.
  Pass: Error_Response, 94 20 FF, Invalid_Parameter.

5.3.2) Add path, invalid attribute and class

- Send an Add_AckData_Path request with invalid attribute.
  Pass: Error_Response, 94 20 FF, Invalid_Parameter.
- Send an Add_AckData_Path request with invalid class.
  Pass: Error_Response, 94 20 FF, Invalid_Parameter.

5.3.3) Remove path, Invalid Instance

- Send a Remove_AckData_Path request with invalid instance Id.
  Pass: Error_Response, 94 20 FF, Invalid_Parameter.

6) Behavior tests.

- Send an Allocate request, choice = COS or Cyclic.

Note: Steps 6.1 through 6.15 are done for both Change of State and Cyclic connections.

6.1) Ack Handler Attribute, I/O Cxn State = Configuring

- Send a Get_Attribute_Single request to the Ack_Timer attribute.
  Pass: Service_Success, value = 16.
- Send a Get_Attribute_Single request to the Retry_Limit attribute.
  Pass: Service_Success, value = 1.
- Send a Get_Attribute_Single request to the COS_Producing_Connection_Instance attribute.
  Pass: Service_Success, value = 4.

The Optional attributes are verified only if they are implemented.

- Send a Get_Attribute_Single request to the Ack_List_Size attribute.
  Pass: Success_Response, value = 0.
- Send a Get_Attribute_Single request to the Ack_List attribute.
  Pass: Success_Response, value = NULL.
- Send a Get_Attribute_Single request to the Data_With_Ack_Path_List_Size attribute.
  Pass: Success_Response, value = 0.
- Send a Get_Attribute_Single request to the Data_With_Ack_Path_List attribute.
  Pass: Success_Response, value = NULL.

6.2) Ack_Timer rounding test.

- Send A Set_Attribute_Single to the I/O connection Ack_Timer, value = 10..20 ms.
  Pass: Success_Response with Ack_Timer value greater than or equal to 10..20.
- Send A Set_Attribute_Single to the I/O connection Ack_Timer, value = 125..135 ms.
  Pass: Success_Response with Ack_Timer value greater than or equal to 125..135.
- Send A Set_Attribute_Single to the I/O connection Ack_Timer, value = 250..260 ms.
  Pass: Success_Response with Ack_Timer value greater than or equal to 250..260.
- Send A Set_Attribute_Single to the I/O connection Ack_Timer, value = 2045..2055 ms.
  Pass: Success_Response with Ack_Timer value greater than or equal to 2045..2055.
  Send A Set_Attribute_Single to the I/O connection Ack_Timer, value = 4090..5000 ms.
  Pass: Success_Response with Ack_Timer value greater than or equal to 4090..5000.
- Send A Set_Attribute_Single to the I/O connection Ack_Timer, value = 32760..32770 ms.
  Pass: Success_Response with Ack_Timer value greater than or equal to 32760..32770.
- Send A Set_Attribute_Single to the I/O connection Ack_Timer, value = 65525..65535 ms.
  Pass: Success_Response with Ack_Timer value greater than or equal to 65525..65535.
  Note: If overflow occurs, the maximum value is accepted. Stop Ack_Timer rounding test.
- Send a Set_Attribute_Single request to the Ack_Timer attribute, value = 50 ms.
  Pass: Service_Success, actual Ack_Timer value.

6.3) COS_Producing_Connection_Instance attribute, State = Inactive

- If the COS_Producing_Connection_Instance attribute is settable, send a Set_Attribute_Single request to the COS_Producing_Connection attribute, value = I/O connection instance Id.
  Pass: Success_Response.

6.4) Initial I/O Data Production Test

- Send a Set_Attribute_Single request to the EPR attribute, value = 500 ms. If necessary, send an Apply_Attributes request to transition the I/O connection to the Established state.
- Consume the device I/O data production.
  Pass: device produces I/O data immediately.
- Send an Acknowledge I/O message, Connection Identifier = Consumed Connection Identifier.

6.5) Set COS_Producing_Connection_Instance attribute, State = Active

- If the COS_Producing_Connection_Instance attribute is settable, send a Set_Attribute_Single request to the COS_Producing_Connection attribute, value = I/O connection instance Id.
  Pass: Error_Response, 9410 FF, Device State Conflict.

6.6) I/O Production on Transmission Trigger Timer Expiration

- Start an internal test timer.
- Consume the device I/O data production based on the actual EPR returned from Set EPR.
- Determine time interval of the I/O data transmission.
  Info: Time value for I/O data produced upon expiration of the Transmission Trigger Timer.
- Send an Acknowledge I/O message, Connection Identifier = Consumed Connection Identifier.

6.7) Ack, Retry Counter Reset

- Check the I/O input buffer.

Pass: device produces no I/O data. The Acknowledgment data was recognized.

6.8) Send invalid Ack, then valid Ack

- Consume the device I/O data production based on the actual EPR returned from Set EPR.
- Determine time interval of the I/O data transmission.
  Pass: device produces I/O data upon expiration of the Transmission Trigger Timer.
- Send an Acknowledge I/O message, size = Consumed Connection Size + 1 byte.
- Start an internal test timer.
- Check the I/O input buffer.
- Determine time interval of the I/O data transmission.
  Pass: device produces I/O data when the Ack_Timer expires.
- Send an Acknowledge I/O message, size = Consumed Connection Size.
- Start an internal test timer.
- Check the I/O input buffer.
  Pass: device produces no I/O data. The Acknowledgment data was recognized.

6.9) Acknowledge Handler Attributes, State = Active

- Send a Get_Attribute_Single request to the Ack_Timer attribute.
  Pass: Service_Success, value = actual Ack_Timer value.
- Send a Get_Attribute_Single request to the Retry_Limit attribute.
  Pass: Service_Success, value = 1.
- Send a Get_Attribute_Single request to the COS_Producing_Connection_Instance attribute.
  Pass: Service_Success, value = 4.

The Optional attributes are verified only if they are implemented.

- Send a Get_Attribute_Single request to the Ack_List_Size attribute.
  Pass: Success_Response, value = 1.
- Send a Get_Attribute_Single request to the Ack_List attribute.
  Pass: Success_Response, value = 4, or Dynamically created instance Id.
- Send a Get_Attribute_Single request to the Data_With_Ack_Path_List_Size attribute.
  Pass: Success_Response, value = 0.
- Send a Get_Attribute_Single request to the Data_With_Ack_Path_List attribute.
  Pass: Success_Response, value = NULL.

6.10) Retry Limit Test. Test is done only if the Retry_Limit is Settable.

- Consume the device I/O data production based on the actual EPR returned from Set EPR.
- Consume the device I/O data production, retry
- Send a Set_Attribute_Single request to the Retry_Limit, value = 2.
- Consume the device I/O data production based on the actual EPR returned from Set EPR.
- Consume the device I/O data production, retry
- Consume the device I/O data production, 2nd retry
- Send an Acknowledge I/O message, size = Consumed Connection Size.
- Start an internal test timer.

- Check the I/O input buffer.

  Pass: device produces no I/O data. The Acknowledgment data was recognized.
- Send a Set_Attribute_Single request to the Retry_Limit, value = 1.

6.11) I/O Inactivity Timer Expiration

- Consume the device I/O data production based on the actual EPR returned from Set EPR.

  Pass: device produces I/O data, for at least 3 Transmission Trigger Timer expiration events
- Consume the device I/O data production, retry
  Pass: device produces I/O data retry for each Acknowledge TimeOut event.
- Send a Get_Attribute_Single request to the I/O connection State attribute.
  Pass: State = TimedOut.

6.12) Acknowledge Handler attributes, I/O Connection State = TimedOut

- Verify Acknowledge Handler attributes according to step 6.8.

6.13) Reset Connection, Get First I/O data

- Send a Reset request to the I/O connection State attribute.
- Consume the device I/O data production.
  Pass: device produces I/O data immediately.
- Send an Acknowledge I/O message, size = Consumed Connection Size.
- Start an internal test timer.
- Check the I/O input buffer.

  Pass: device produces no I/O data. The Acknowledgment data was recognized.
- Verify Acknowledge Handler attributes according to step 6.8.
- Consume the I/O data and retries according to step 6.9
- Send a Get_Attribute_Single request to the I/O connection State attribute.
  Pass: State = TimedOut.

6.14) Re-Allocate the Connection, Get I/O data

- Send an Allocate request, choice = I/O connection.
- Send a Get_Attribute_Single request to the I/O connection State attribute.
  Pass: State = Configuring.
- Send a Set_Attribute_Single request to the EPR attribute, value = 500 ms. If necessary, send an Apply_Attributes request to transition the I/O connection to the Established state.
- Consume the device I/O data production.
  Pass: device produces I/O data immediately.
- Send an Acknowledge I/O message, Connection Identifier = Consumed Connection Identifier.
- Send a Get_Attribute_Single request to the I/O connection State attribute.
  Pass: State = Established.
- Verify Acknowledge Handler attributes according to step 6.8.

6.15) Acknowledge Handler Attributes, Connection Deleted.

- Send a Get_Attribute_Single request to the Ack_Timer attribute.
  Pass: Service_Success, value = actual Ack_Timer value, or

Pass: Error_Response, 9416 ff Object_Does_Not_Exist, if Ack Handler deleted. Skip step 6.15.

- Send a Get_Attribute_Single request to the Retry_Limit attribute.
  Pass: Service_Success, value = 1.
- Send a Get_Attribute_Single request to the COS_Producing_Connection_Instance attribute.
  Pass: Service_Success, value = 4.

The Optional attributes are verified only if they are implemented.

- Send a Get_Attribute_Single request to the Ack_List_Size attribute.
  Pass: Success_Response, value = 0.
- Send a Get_Attribute_Single request to the Ack_List attribute.
  Pass: Success_Response, value = NULL.
- Send a Get_Attribute_Single request to the Data_With_Ack_Path_List_Size attribute.
  Pass: Success_Response, value = 0.
- Send a Get_Attribute_Single request to the Data_With_Ack_Path_List attribute.
  Pass: Success_Response, value = NULL.

6.16) Dynamic Acknowledge Handler Test.

- Create and configure a Dynamic Ack Handler Instance.
- Create and configure a Dynamic I/O connection for Change of state behavior.
- Set the Consumed Connection Path to the created Ack Handler Instance.

6.17) Add_AckData_Path Test. If Add_AckData_Path is implemented

- Add_AckData_Path to add 2 path list members for Ack with Data.

Verify the Dynamic Ack Handler behavior according to steps 6.1 through 6.15.

6.18) Delete the Dynamic Acknowledge Handler Instance.

Pass: Success_Response.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.30 Overload Object Test x2C

This section defines the Overload object conformance test. This test is required for all Motor Starter type devices (device type 0x16) and Overload type devices (device type 3).

Functional Description:

This test verifies the:

1) Class attributes access rules for the Overload object.
2) Instance attributes access rules for the Overload object.
3) Common service implementations for the class and instance.
4) Object-specific and Reserved services for the class and instance.
5) Conformance when incorrect data is used to access attributes and services.

5.1) response when Set_Attribute_Single is requested with invalid data.
5.2) response when Set_Attribute_Single is requested with not enough data.
5.3) response when Set_Attribute_Single is requested with too much data.

6) Object behavior accessible from the network.

Procedure Definition:

- Initialization
  Log the object name and object and object test software revision and version identifier.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 5000ms.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(6..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(3..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
NumAttr(01)	[UINT(1..65535), Attribute_Not_Supported]
Attributes (02)	[USINT[NumbAttr], Attribute_Not_Supported]
TripFLCSet (03)	[INT, Attribute_Not_Supported]
TripClass (04)	[USINT, Attribute_Not_Supported]
AvgCurrent (05)	[INT, Attribute_Not_Supported]
%PhImbal (06)	[USINT, Attribute_Not_Supported]
%Thermal (07)	[USINT, Attribute_Not_Supported]
CurrentL1 (08)	[INT, Attribute_Not_Supported]
CurrentL2 (09)	[INT, Attribute_Not_Supported]
CurrentL3 (10)	[INT, Attribute_Not_Supported]
GroundCurrent (11)	[INT, Attribute_Not_Supported]
CurrentScale (12)	[SINT, Attribute_Not_Supported]
Undefined (13..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
NumAttr(01)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes (02)	[Attribute_Not_Settable, Attribute_Not_Supported]
TripFLCSet (03)	[Success_Response, Attribute_Not_Supported]
TripClass (04)	[Success_Response, Attribute_Not_Supported]
AvgCurrent (05)	[Attribute_Not_Settable, Attribute_Not_Supported]
%PhImbal (06)	[Attribute_Not_Settable, Attribute_Not_Supported]
%Thermal (07)	[Attribute_Not_Settable, Attribute_Not_Supported]
CurrentL1 (08)	[Attribute_Not_Settable, Attribute_Not_Supported]
CurrentL2 (09)	[Attribute_Not_Settable, Attribute_Not_Supported]
CurrentL3 (10)	[Attribute_Not_Settable, Attribute_Not_Supported]
vGroundCurrent (11)	[Attribute_Not_Settable, Attribute_Not_Supported]
CurrentScale (12)	[Success_Response, Attribute_Not_Supported]
Undefined (13..99)	[Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] if no attributes are supported
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (Code)	[Expected Response]
Reserved (00)	[Service_Not_Supported (x08)]
Reserved (01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Supported, Attribute_Not_Settable (x0E)]
Reserved (17..20)	[Service_Not_Supported]
Restore (21)	[Success_Response, Service_Not_Supported, Object_State_Conflict(x0C)]
Save (22)	[Success_Response, Service_Not_Supported]
Reserved (23..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) Test the implemented attributes accessible by Set_Attribute_Single using invalid data.

- attribute 4, TripClass = 201.
  Pass: Error_Response, 94 09 FF, Invalid_Attribute_Value

5.2) Test the implemented attributes accessible by Set_Attribute_Single using insufficient data.

- Request a Set_Attribute_Single, instance 1. Use xFF for each byte shown for each attribute.
- attribute 3, TripFLCSet with 1 byte of data.
- attribute 4, TripClass with 0 bytes of data.
- attribute 12, CurrentScale with 0 bytes of data.
  Pass: Error_Response, 94 13 FF, Not_Enough_Data for all responses

5.3) Test the implemented attributes accessible by Set_Attribute_Single using too much data.

- Request a Set_Attribute_Single, instance 1. Use xFF for each byte shown for each attribute.
- attribute 3, TripFLCSet with 4 bytes of data.
- attribute 4, TripClass with 2 bytes of data.
- attribute 12, CurrentScale with 2 bytes of data.
  Pass: Error_Response, 94 15 FF, Too_Much_Data for all responses

6) Behavior tests.

None. The Overload object includes attributes that in general effect behavior that is associated with the physical system that the device is connected to. This type of behavior is not observable from the network interface.

6.1) If instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.31 SoftStart Object Test x2D

This section defines the Softstart object conformance test. This test is required for all Softstarter type devices (device type 0x17).

Functional Description:

This test verifies the:
1) Class attributes access rules for the Softstart object.
2) Instance attributes access rules for the Softstart object.
3) Common service implementations for the class and instance.
4) Object-specific and Reserved services for the class and instance.
5) Conformance when incorrect data is used to access attributes and services.
5.1) response when Set_Attribute_Single is requested with invalid data.
5.2) response when Set_Attribute_Single is requested with not enough data.
5.3) response when Set_Attribute_Single is requested with too much data.
6) Object behavior accessible from the network.

Procedure Definition:

- Initialization
  The Example object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 5000ms.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported]
Num_Instances (03)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(6..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(4..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (02..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
NumAttr(01)	[UINT(4..255), Attribute_Not_Supported]
Attributes (02)	[USINT[NumAttr], Attribute_Not_Supported]
AtReference (03)	[BOOL]
StartMode(04)	[USINT]
StopMode (05)	[USINT, Attribute_Not_Supported]
RampMode (06)	[USINT, Attribute_Not_Supported]
RampTime1 (07)	[UINT, Attribute_Not_Supported]
InitialVoltage1(08)	[USINT, Attribute_Not_Supported]
RampTime2 (09)	[UINT, Attribute_Not_Supported]
InitialVoltage2 (10)	[USINT, Attribute_Not_Supported]
Rotation (11)	[BOOL, Attribute_Not_Supported]
KickStart (12)	[BOOL, Attribute_Not_Supported]
KickStartTime (13)	[USINT, Attribute_Not_Supported]
KickStartVoltage (14)	[UINT, Attribute_Not_Supported]
EnergySaver (15)	[BOOL, Attribute_Not_Supported]
DecelTime (16)	[UINT, Attribute_Not_Supported]
CurrentLimitSet (17)	[UINT, Attribute_Not_Supported]
BrakingCurrentSet (18)	[UINT, Attribute_Not_Supported]
Undefined (19..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
NumAttr(01)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes (02)	[Attribute_Not_Settable, Attribute_Not_Supported]
AtReference (03)	[Attribute_Not_Settable]
StartMode(04)	[Success_Response]
StopMode (05)	[Success_Response, Attribute_Not_Supported]
RampMode (06)	[Success_Response, Attribute_Not_Supported]
RampTime1 (07)	[Success_Response, Attribute_Not_Supported]
InitialVoltage1(08)	[Success_Response, Attribute_Not_Supported]
RampTime2 (09)	[Success_Response, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
InitialVoltage2 (10)	[Success_Response, Attribute_Not_Supported]
Rotation (11)	[Success_Response, Attribute_Not_Supported]
KickStart (12)	[Success_Response, Attribute_Not_Supported]
KickStartTime (13)	[Success_Response, Attribute_Not_Supported]
KickStartVoltage (14)	[Success_Response, Attribute_Not_Supported]
EnergySaver (15)	[Success_Response, Attribute_Not_Supported]
DecelTime (16)	[Success_Response, Attribute_Not_Supported]
CurrentLimitSet (17)	[Success_Response, Attribute_Not_Supported]
BrakingCurrentSet (18)	[Success_Response, Attribute_Not_Supported]
Undefined (19..99)	[Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] if no attributes are supported
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (Code)	[Expected Response]
Reserved (00)	[Service_Not_Supported (x08)]
Reserved (01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable (x0E) , Attribute_Not_Supported (x14)]
Reserved (17..20)	[Service_Not_Supported]
Restore (21)	[Success_Response, Service_Not_Supported, Object_State_Conflict(x0C)]
Save (22)	[Success_Response, Service_Not_Supported]
Reserved (23..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.

Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) Test the implemented attributes accessible by Set_Attribute_Single using invalid data.

- attribute 11, Rotation = 2.
- attribute 12, KickStart = 2.
- attribute 15, EnergySaver = xFF.
  Pass: Error_Response, 94 09 FF, Invalid_Attribute_Value

5.2) Test the implemented attributes accessible by Set_Attribute_Single using insufficient data.

- Request a Set_Attribute_Single, instance 1. Use xFF for each byte shown for each attribute.
- attribute 4, StartMode with 0 bytes of data.
- attribute 5, StopMode with 0 bytes of data.
- attribute 6, RampMode with 0 bytes of data.
- attribute 7, RampTime1 with 1 byte of data.
- attribute 8, InitialVoltage1 with 0 bytes of data.
- attribute 9, RampTime2 with 1 byte of data.
- attribute 10, InitialVoltage2 with 0 bytes of data.
- attribute 11, Rotation with 0 bytes of data.
- attribute 12, KickStart with 0 bytes of data.
- attribute 13, KickStartTime with 0 bytes of data.
- attribute 14, KickStartVoltage with 1 byte of data.
- attribute 15, EnergySaver with 0 bytes of data.
- attribute 16, DecelTime with 1 byte of data.
- attribute 17, CurrentLimitSet with 1 byte of data.
- attribute 18, BrakingCurrentSet with 1 byte of data.
  Pass: Error_Response, 94 13 FF, Not_Enough_Data for all responses

5.3) Test the implemented attributes accessible by Set_Attribute_Single using too much data.

- Request a Set_Attribute_Single, instance 1. Use xFF for each byte shown for each attribute.

- attribute 4, StartMode with 2 bytes of data.
- attribute 5, StopMode with 2 bytes of data.
- attribute 6, RampMode with 2 bytes of data.
- attribute 7, RampTime1 with 4 bytes of data.
- attribute 8, InitialVoltage1 with 2 bytes of data.
- attribute 9, RampTime2 with 4 bytes of data.
- attribute 10, InitialVoltage2 with 2 bytes of data.
- attribute 11, Rotation with 2 bytes of data.
- attribute 12, KickStart with 2 bytes of data.
- attribute 13, KickStartTime with 2 bytes of data.
- attribute 14, KickStartVoltage with 4 bytes of data.
- attribute 15, EnergySaver with 2 bytes of data.
- attribute 16, DecelTime with 4 bytes of data.
- attribute 17, CurrentLimitSet with 4 bytes of data.
- attribute 18, BrakingCurrentSet with 4 bytes of data.
  Pass: Error_Response, 94 15 FF, Too_Much_Data for all responses

6) Behavior Test
None. The Softstart object includes attributes that in general effect behavior that is associated with the physical system that the softstart is connected to. This type of behavior is not observable from the network interface.

6.1) If instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.32 Selection Object Test x2E

This section defines the Selection object conformance test. This test is required when the Selection object is implemented.

Functional Description:

This test verifies the:

1) Class attributes access rules for the Selection object.
2) Instance attributes access rules for the Selection object.
3) Common service implementations for the class and instance.
4) Object-specific and Reserved services for the class and instance.
5) Conformance when incorrect data is used to access attributes and services.

5.1) response when Set_Attribute_Single is requested with invalid data.
5.2) response when services are requested with invalid data.

6) Object behavior accessible from the network.

6.1) Dynamic Tests - Create/Add/Remove/Delete
6.2) Algorithm Type Tests
6.2.1) Hot Backup Test
6.2.2) First Arrival Test
6.2.3) Last Arrival Test
6.2.4) Programmable Data Flow Test
6.2.4.1) Programmable Data Flow Test, Process Control Valve

Procedure Definition:

- Initialization
  Log the object name and object and object test software revision and version identifier.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 5000 ms.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT (2]
Max_Instance (02)	[UINT (1..65535), Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), size is first UINT, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), size is first UINT, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(6..199), Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(3..199), Attribute_Not_Supported]
Reserved (08)	[Attribute_Not_Supported]
Max_Number_of_Instances (09)	[UINT, Attribute_Not_Supported]
Undefined (10..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access

attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
State (01)	[USINT(0..2)]
Max_Destinations (02)	[UINT] ***Note 1***
Number_of_Destinations (03)	[UINT If Max_Destinations > 0, Attribute_Not_Supported] ***Note 1***
Destination_List (04)	[[USINT, EPATH]Number_of_Destinations] If Max_Destinations > 0, Attribute_Not_Supported] ***Note 1***
Max_Sources (05)	[UINT] ***Note 1***
Number_of_Sources (06)	[UINT If Max_Sources > 0, Attribute_Not_Supported] ***Note 1***
Source_List (07)	[[USINT, EPATH]Number_of_Sources] If Max_Sources > 0, Attribute_Not_Supported] ***Note 1***
Source_Used (08)	[UINT, Attribute_Not_Supported] ***Note 1***
Source_Alarm (09)	[[BOOL] Max_Sources]], Attribute_Not_Supported]
Algorithm_Type (10)	[USINT(0..4, 100..199), Attribute_Not_Supported] ***Note 1***
Detection_Count (11)	[USINT(2..255) If Algorithm_Type supported, Attribute_Not_Supported]
Selection_Period (12)	[UINT(1..65535)] if Algorithm_Type supported, Attribute_Not_Supported]
Object_Source_List (13)	[[USINT, EPATH]Number_of_Sources], Attribute_Not_Supported] ***Note 1***
Destination_Used (14)	[UINT(0.. Number_of_Destinations)], Attribute_Not_Supported] ***Note 1***
Input_Data_Value (15)	[Input Data Type, If Max_Sources = 0, Attribute_Not_Supported]
Output_Data_Value (16)	[Output Data Type, If Max_Destinations > 0, Attribute_Not_Supported]
Undefined (17..99)	[Attribute_Not_Supported]

Note 1: The following values are defined and limited by the device profile

Process Control Valve (1D)	Instance 1
Attribute Name (ID)	**[Expected Values, Responses]**
Max_Destinations (02)	[UINT(2)]
Number_of_Destinations (03)	[UINT(2)]

Process Control Valve (1D)	**Instance 1**
Attribute Name (ID)	**[Expected Values, Responses]**
Destination_List (04)	[06 20 33 24 01 30 06 06 20 33 24 02 30 06]
Max_Sources (05)	[UINT(0)]
Number_of_Sources (06)	[UINT(0)]
Source_List (07)	[Attribute_Not_Supported]
Source _Used (08)	[UINT(0)]
Algorithm_Type (10)	[USINT(4)]
Object_Source_List (13)	[00]
Destination_Used (14)	[UINT(0..2)]

Process Control Valve (1D)	**Instance 2**
Attribute Name (ID)	**[Expected Values, Responses]**
Max_Destinations (02)	[UINT(1)]
Number_of_Destinations (03)	[UINT(1)]
Destination_List (04)	[06 20 32 24 01 30 06]
Max_Sources (05)	[UINT(2)]
Number_of_Sources (06)	[UINT(2)]
Source_List (07)	[Attribute_Not_Supported]
Source _Used (08)	[UINT(1,2)]
Algorithm_Type (10)	[USINT(4)]
Object_Source_List (13)	[06 20 33 24 01 30 09 06 20 33 24 02 30 09]
Destination_Used (14)	[UINT(1)]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	**[Expected Values, Responses]**
Undefined (00)	[Attribute_Not_Supported (x14)]
State (01)	[Attribute_Not_Settable (x0E)]
Max_Destinations (02)	[Attribute_Not_Settable]
Number_of_Destinations (03)	[Attribute_Not_Settable, Attribute_Not_Supported]
Destination_List (04)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported] ***Note 2***
Max_Sources (05)	[Attribute_Not_Settable]
Number_of_Sources (06)	[Attribute_Not_Settable, Attribute_Not_Supported]
Source_List (07)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported] ***Note 2***
Source_Used (08)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported] ***Note 2***
Source_Alarm (09)	[Attribute_Not_Settable, Attribute_Not_Supported]
Algorithm_Type (10)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported] ***Note 2***

Attribute Name (ID)	[Expected Values, Responses]
Detection_Count (11)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Selection_Period (12)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Object_Source_List (13)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported] ***Note 2***
Destination_Used (14)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported] ***Note 2***
Input_Data_Value (15)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Output_Data_Value (16)	[Attribute_Not_Settable, Attribute_Not_Supported]
Undefined (17..99)	[Attribute_Not_Supported]

Note 2: The following values are defined and limited by the device profile

Process Control Valve (1D)	
Attribute Name (ID)	**[Expected Values, Responses]**
Destination_List (04)	[Attribute_Not_Settable]
Algorithm_Type (07)	[Attribute_Not_Settable]
Source_Used (08)	[Attribute_Not_Settable]
Algorithm_Type (10)	[Attribute_Not_Settable]
Object_Source_List (13)	[Attribute_Not_Settable]
Destination_Used (14)	[Attribute_Not_Settable]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
(01..4)	[Service_Not_Supported]
Reset (05)	[Success_Response, Service_Not_Supported]
Start (06)	[Success_Response, Service_Not_Supported]
Stop (07)	[Success_Response, Service_Not_Supported]
Create (08)	[Success_Response, Service_Not_Supported]
Delete (09)	[Success_Response, Service_Not_Supported]
(10..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response]
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (Code)	[Expected Response]
Reserved (00)	[Service_Not_Supported (x08)]
(01..4)	[Service_Not_Supported]
Reset (05)	[Success_Response, Service_Not_Supported]
Start (06)	[Success_Response, Service_Not_Supported]

Service Name (Code)	[Expected Response]
Stop (07)	[Success_Response, Service_Not_Supported]
Create (08)	[Service_Not_Supported]
Delete (09)	[Success_Response, Service_Not_Supported]
(10..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Supported, Attribute_Not_Settable (x0E)]
Reserved (17)	[Service_Not_Supported]
Get_Member (18)	[Success_Response, Service_Not_Supported]
Set_Member (19)	[Success_Response, Service_Not_Supported]
Insert_Member (1A)	[Success_Response, Service_Not_Supported]
Remove_Member (1B)	[Success_Response, Service_Not_Supported]
Reserved (1C..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) Test attributes accessible by Set_Attribute_Single using invalid data.

- attribute 4, Destination_List and attribute 13, Object_Source_List = Invalid Object Class.
  Pass: Error_Response, 94 05 FF, Path Destination Unknown

- attribute 8, Source_Used = Max_Sources + 1.
  Pass: Error_Response, 94 09 FF, Invalid_Attribute_Value
- attribute 10, Algorithm_Type = 5..99, 200..255.
  Pass: Error_Response, 94 09 FF, Invalid_Attribute_Value
- attribute 11, Destination_Count = 0,1.
  Pass: Error_Response, 94 09 FF, Invalid_Attribute_Value
- attribute 12, Selection_Period = 0.
  Pass: Error_Response, 94 09 FF, Invalid_Attribute_Value
- attribute 14, Destination_Used = Max_Destinations + 1.
  Pass: Error_Response, 94 09 FF, Invalid_Attribute_Value

5.2) Services Test, for implemented services.

- List Attribute = Destination_List, or Source_List.
- Request a Get_Member, List Attribute, MemberId = 0.
  Pass: Error_Response, 94 28 FF, Invalid Member Id
- Request a Set_Member, List Attribute, MemberId = 127.
  Pass: Error_Response, 94 28 FF, Invalid Member Id
- Request a Set_Member, List Attribute, MemberId = 0.
  Pass: Error_Response, 94 28 FF, Invalid Member Id
- Request a Insert_Member, List Attribute, MemberId = 0.
  Pass: Error_Response, 94 28 FF, Invalid Member Id
- Request a Remove_Member, List Attribute, MemberId = 0.
  Pass: Error_Response, 94 28 FF, Invalid Member Id

6) Behavior tests.

6.1) Dynamic Tests - Create/Add/Remove/Delete

- Tests to be defined.

6.2) Algorithm Type Test

6.2.1) Hot Backup Test

- Tests to be defined.

6.2.2) First Arrival Test

- Tests to be defined.

6.2.3) Last Arrival Test

- Tests to be defined.

6.2.4) Programmable Data Flow Test

- Tests to be defined.

6.2.4.1) Programmable Data Flow Test, Process Control Valve

- Get Attribute S-Analog Actuator, instance 1, attribute 6
  Pass: Success_Response, = saa_value
- Send I/O data to assembly instance 7, Setpoint = saa_value + 10, instance = 1
- Get Attribute S-Single Stage Controller, instance 1, attribute 9, Control_Variable
  Pass: Success_Response Value = saa_value + 10

- Get Attribute S-Analog Actuator, instance 1, attribute 6, Value
  Pass: Success_Response Value = saa_value + 10
- Get Attribute Selection, instance 1, attribute 8, Source_Used
  Pass: Success_Response Value = 1
- Send I/O data to assembly instance 7, Setpoint = saa_value + 10, instance = 2
- Get Attribute S-Single Stage Controller, instance 2, attribute 9, Control_Variable
  Pass: Success_Response Value = saa_value + 10
- Get Attribute S-Analog Actuator, instance 1, attribute 6, Value
  Pass: Success_Response Value = saa_value + 10
- Get Attribute Selection, instance 1, attribute 8, Source_Used
  Pass: Success_Response Value = 2

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.33 S-Device Supervisor Object Test x30

This section defines the conformance test for the S-Device Supervisor object. This test is required when the S-Device Supervisor object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the S-Device Supervisor object.

2) Instance attributes access rules for the S-Device Supervisor object.

3) Common Service implementations for class and instance.

4) Object–specific and Reserved service implementations for class and instance.

5) Conformance when incorrect data is used to access attributes and services.

5.1) response when Set_Attribute_Single is requested with invalid data.

5.2) response when Perform_Diagnostics Service is requested with invalid data.

6) Object behavior accessible from the network.

6.1) Exception_Status attribute behavior.
6.2) Exception_Status and Exception_Detail attribute behavior.
6.3) Alarm_Enable and Warning_Enable attribute behavior.
6.4) Clock_Behavior and Time attributes behavior.
6.5) non-volatile attributes behavior.
6.6) implementation of the SHORT_STRING attributes.

6.7) state transition behavior.

6.8) Identity object Reset Interaction behavior.

6.9) Device Profile specific behavior.

6.10) implementation of the Attributes_List attribute.

Procedure Definition

- Initialization
  The S-Device Supervisor object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection, EPR = 2500 ms.

1) Class attributes access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Max_Class_Attribute_Id (06)	[UINT(6..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(16..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..98)	[Service_Not_Supported, Attribute_Not_Supported]
Subclass (99)	[UINT (0), Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attributes access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[USINT, Attribute_Not_Supported] ***Note 1***
Attributes_List (02)	[USINT[Number_of_Attributes], Attribute_Not_Supported]
Device_Type (03)	[SHORT_STRING, 8 Characters max] ***Note 1***
SEMI_Std_Revision_Level (04)	["E54-0997"]
Manufacturer_Name (05)	[SHORT_STRING, 20 Characters max]
Manufacturer_Model_No (06)	[SHORT_STRING, 20 Characters max]
Software_Revision (07)	[SHORT_STRING, 6 Characters max]
Hardware_Revision (08)	[SHORT_STRING, 6 Characters max]
Manufacturer_Serial_No (09)	[SHORT_STRING, 30 Characters max, Attribute_Not_Supported]
Device_Configuration (10)	[SHORT_STRING, 50 Characters max, Attribute_Not_Supported]
Device_Status (11)	[USINT(0..6, 51..255)]
Exception_Status (12)	[BYTE(0xxxxxxx, 1xxx0xxx)] 0 = must be zero, x = 0 or 1
Exception_Detail_Alarm (13)	[(Common_Except_Sz(USINT(2)), BYTE[2*] , Device_Except_Sz(USINT(0..**)), BYTE[Device_Except_Sz] , Mfg_Except_Sz(USINT(0..**)), BYTE[Mfg_Except_Sz]), Attribute_Not_Supported] Only if Exception_Status bit 7 = 0 * = [0x0xxxxx, 0xxxxxxx] 0 = must be zero, x = 0 or 1 ** = ***Note 1***

Attribute Name (ID)	**[Expected Values, Responses]**
Exception_Detail_Warning (14)	[(Common_Except_Sz(USINT(2)), BYTE[2*] , Device_Except_Sz(USINT(0..255)), BYTE[Device_Except_Sz] , Mfg_Except_Sz(USINT(0..255)), BYTE[Mfg_Except_Sz]) , Attribute_Not_Supported] Only if Exception_Status bit 7 = 0 * = [0x0xxxxx, 0xxxxx0x] 0 = must be zero, x = 0 or 1 ** = ***Note 1***
Alarm_Enable (15)	[BOOL(1)] Default = 1
Warning_Enable (16)	[BOOL(1)] Default = 1
Time (17)	[DATE_AND_TIME (UDINT, UINT), Attribute_Not_Supported]
Clock_Behavior (18)	[USINT(0, 1, 2) , Attribute_Not_Supported]
Last_Maintenance_Date (19)	[DATE, Attribute_Not_Supported]
Next_Maintenance_Date (20)	[DATE, Attribute_Not_Supported]
Maintenance_Timer (21)	[INT, Attribute_Not_Supported]
Maintenance_Warning_Enable (22)	[BOOL(1), Default = 1* , Attribute_Not_Supported] * = Required if Maintenance_Timer is implemented
Time (23)	[UDINT, Attribute_Not_Supported]
Endpoint (24)	[BOOL, Attribute_Not_Supported]
Recipe (25)	[UINT, Attribute_Not_Supported]
Undefined (26..98)	[Attribute_Not_Supported] ***Note 2***
Subclass (99)	[UINT (0, 1), Attribute_Not_Supported] ***Note 1***

Note 1: Values for these attributes are defined and limited by the Device Profile

Mass Flow Controller (1A)	
Attribute Name (ID)	**[Expected Values]**
Number_of_Attributes (01)	(10..26 + vendor specific)
Device_Type (03)	["MFC", "MFM"]
Exception_Detail_Alarm (13)	Device_Except_Sz = 1, Mfg_Except_Sz = 1
Exception_Detail_Warning (14)	Device_Except_Sz = 1, Mfg_Except_Sz = 1
Subclass (99)	[UINT (0), Attribute_Not_Supported]

Pressure/Vacuum Guage (1C)	
Attribute Name (ID)	**[Expected Values]**
Number_of_Attributes (01)	(10..26 + vendor specific)
Device_Type (03)	["VG", "CG", "MG"]
Exception_Detail_Alarm (13)	Device_Except_Sz = (3..31), Mfg_Except_Sz = 1
Exception_Detail_Warning (14)	Device_Except_Sz = (3..31), Mfg_Except_Sz = 1
Subclass (99)	[UINT (0), Attribute_Not_Supported]

Process Control Valve (1D)	
Attribute Name (ID)	**[Expected Values]**
Number_of_Attributes (01)	(10..26 + vendor specific)
Device_Type (03)	["PCV"]

Process Control Valve (1D)	
Attribute Name (ID)	**[Expected Values]**
Exception_Detail_Alarm (13)	Device_Except_Sz = 5, Mfg_Except_Sz = 1
Exception_Detail_Warning (14)	Device_Except_Sz = 5, Mfg_Except_Sz = 1
Subclass (99)	[UINT (0), Attribute_Not_Supported]

Residual Gas Analyser (1E)	
Attribute Name (ID)	**[Expected Values]**
Number_of_Attributes (01)	(10..26 + vendor specific)
Device_Type (03)	[“RGA”]
Exception_Detail_Alarm (13)	Device_Except_Sz = 4, Mfg_Except_Sz = 1
Exception_Detail_Warning (14)	Device_Except_Sz = 4, Mfg_Except_Sz = 1
Subclass (99)	[UINT (0), Attribute_Not_Supported]

DC Power Genreator (1F)	
Attribute Name (ID)	**[Expected Values]**
Number_of_Attributes (01)	(10..26 + vendor specific)
Device_Type (03)	[“DCG”]
Exception_Detail_Alarm (13)	Device_Except_Sz = 4, Mfg_Except_Sz = 1
Exception_Detail_Warning (14)	Device_Except_Sz = 4, Mfg_Except_Sz = 1
Subclass (99)	[UINT (0), Attribute_Not_Supported]

RF Power Genreator (20 hex)	
Attribute Name (ID)	**[Expected Values]**
Number_of_Attributes (01)	(10..26 + vendor specific)
Device_Type (03)	[“RFG”]
Exception_Detail_Alarm (13)	Device_Except_Sz = 6, Mfg_Except_Sz = 1
Exception_Detail_Warning (14)	Device_Except_Sz = 6, Mfg_Except_Sz = 1
Subclass (99)	[UINT (0), Attribute_Not_Supported]

Turbomolecular Vacuum Pump (21 hex)	
Attribute Name (ID)	**[Expected Values]**
Number_of_Attributes (01)	(10..26 + vendor specific)
Device_Type (03)	[“Turbo Pump”]
Exception_Detail_Alarm (13)	Device_Except_Sz = 4, Mfg_Except_Sz = 1
Exception_Detail_Warning (14)	Device_Except_Sz = 4, Mfg_Except_Sz = 1
Subclass (99)	[UINT (1)]

Note 2: Values for these attributes are defined and limited by the Device Profile, Subclass

Vacuum/Pressure Gauge (1C)	SubClass = 5, Hot Cathode Ion Gauge
Attribute Name (ID)	**[Expected Values]**
Energy_Control_Enabled (81)	[BOOL(0), Attribute_Not_Supported]
Joules_to_Deliver (82)	[UDINT(0), Attribute_Not_Supported]
Joules_Remaining (83)	[UDINT(0), Attribute_Not_Supported]
Active_Target_ID (84)	[UINT(0), Attribute_Not_Supported]
Active_Target_Count_Enable(85)	[BOOL(0), Attribute_Not_Supported]
Active_Target_Life(86)	[DINT(0), Attribute_Not_Supported]
Input_Power_On_Time (87)	[UDINT, Attribute_Not_Supported]
Output_Power_On_Time(88)	[UDINT, Attribute_Not_Supported]
Filament_On_Time (89)	[UDINT, Attribute_Not_Supported]
Total_Energy_delivered (90)	[UDINT(1.0), Attribute_Not_Supported]
Power_On_Cycle_Counter (91)	[UDINT, Attribute_Not_Supported]
Output_On_Cycle_Counter (92)	[UDINT(1), Attribute_Not_Supported]
Overtime_Event_Counter (93)	[UDINT, Attribute_Not_Supported]
Reg_Alarm_Event_Counter(94)	[UDINT, Attribute_Not_Supported]
Filament_Power_Enable (95)	[BOOL(0), Attribute_Not_Supported]
Output_Power_Enable(96)	[BOOL(0), Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Num_Attributes (01)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes_List (02)	[Attribute_Not_Settable, Attribute_Not_Supported]
Device_Type (03)	[Attribute_Not_Settable]
SEMI_Std_Revision_Level (04)	[Attribute_Not_Settable]
Manufacturer_Name (05)	[Attribute_Not_Settable]
Manufacturer_Model_No (06)	[Attribute_Not_Settable]
Software_Revision (07)	[Attribute_Not_Settable]
Hardware_Revision (08)	[Attribute_Not_Settable]
Manufacturer_Serial_No (09)	[Attribute_Not_Settable, Attribute_Not_Supported]
Device_Configuration (10)	[Attribute_Not_Settable, Attribute_Not_Supported]
Device_Status (11)	[Attribute_Not_Settable]
Exception_Status (12)	[Attribute_Not_Settable]
Exception_Detail_Alarm (13)	[Attribute_Not_Settable, Attribute_Not_Supported]
Exception_Detail_Warning (14)	[Attribute_Not_Settable, Attribute_Not_Supported]
Alarm_Enable (15)	[Success_Response]
Warning_Enable (16)	[Success_Response]

Attribute Name (ID)	[Expected Responses]
Time (17)	[Success_Response, Attribute_Not_Supported]
Clock_Behavior (18)	[Attribute_Not_Settable, Attribute_Not_Supported]
Last_Maintenance_Date (19)	[Attribute_Not_Settable, Attribute_Not_Supported]
Next_Maintenance_Date (20)	[Attribute_Not_Settable, Attribute_Not_Supported]
Maintenance_Timer (21)	[Attribute_Not_Settable, Attribute_Not_Supported]
Maintenance_Warning_Enable (22)	[Success_Response, Attribute_Not_Supported] * = Required if Maintenance_Timer is implemented
Time (23)	[Attribute_Not_Settable, Attribute_Not_Supported]
Endpoint (24)	[Attribute_Not_Settable, Attribute_Not_Supported]
Recipe (25)	[Success_Response, Attribute_Not_Supported]
Undefined (26..98)	[Attribute_Not_Supported] ***Note 3***
Subclass (99)	[Attribute_Not_Settable, Attribute_Not_Supported]

Note 3: Access for these attributes are defined and limited by the Device Profile, Subclass

Vacuum/Pressure Gauge (1C)	SubClass = 5, Hot Cathode Ion Gauge
Attribute Name (ID)	**[Expected Values]**
Energy_Control_Enabled (81)	[Success_Response, Attribute_Not_Supported]
Joules_to_Deliver (82)	[Success_Response, Attribute_Not_Supported]
Joules_Remaining (83)	[UDINT(0), Attribute_Not_Supported]
Active_Target_ID (84)	[Success_Response, Attribute_Not_Supported]
Active_Target_Count_Enable(85)	[Success_Response, Attribute_Not_Supported]
Active_Target_Life(86)	[Success_Response, Attribute_Not_Supported]
Input_Power_On_Time (87)	[Attribute_Not_Settable, Attribute_Not_Supported]
Output_Power_On_Time(88)	[Attribute_Not_Settable, Attribute_Not_Supported]
Filament_On_Time (89)	[Success_Response, Attribute_Not_Supported]
Total_Energy_delivered (90)	[Attribute_Not_Settable, Attribute_Not_Supported]
Power_On_Cycle_Counter (91)	[Attribute_Not_Settable, Attribute_Not_Supported]
Output_On_Cycle_Counter (92)	[Attribute_Not_Settable, Attribute_Not_Supported]
Overtemp_Event_Counter (93)	[Attribute_Not_Settable, Attribute_Not_Supported]
Reg_Alarm_Event_Counter(94)	[Attribute_Not_Settable, Attribute_Not_Supported]
Filament_Power_Enable (95)	Success_Response, Attribute_Not_Supported]
Output_Power_Enable(96)	[Success_Response, Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]

Service Name (Code)	[Expected Responses]
Get_Attribute_Single (14)	[Success_Response, Service_Not_Supported (if no attributes are supported)]
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (code)	[Expected Responses]
(00..4)	[Service_Not_Supported (x08)]
Reset (05)	[Success_Response]
Start (06)	[Success_Response]
Stop (07)	[Success_Response, Service_Not_Supported]
(08..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Abort (75)	[Success_Response]
Recover (76)	[Success_Response]
undefined (77)	[Service_Not_Supported (x08)]]
Perform_Diagnostics (78)	[Success_Response]
Object–specific Codes (79..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..104)	[Success_Response , Service_Not_Supported] Profile Specific

Service Name (Code)	[Expected Responses]
Reserved Codes (105..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) For each attribute that is implemented with Set access,

- Request a Set_Attribute_Single service, Alarm_Enable = 254.
- Request a Set_Attribute_Single service, Warning_Enable = 254.
- Request a Set_Attribute_Single service, Maintenance_Warning_Enable = 254.
  Pass: For each attribute, Error_Response 94 09 FF, Invalid_Attribute_Value.

5.2) Perform_Diagnostics Service

- Request a Perform_Diagnostics service, TestID = 1..63.
  Pass: For each value, Error_Response 94 20 FF, Invalid_Parameter.
- Request a Perform_Diagnostics service, TestID = values 64..127 not defined in Device Profile.
  Pass: For each value, Error_Response 94 20 FF, Invalid_Parameter.

6) Behavior tests.

6.1) Exception_Status Test

- Request a Get_Attribute_Single, Exception_Status.
  Pass: bit 3 = 0 for MFC

6.2) Exception_Detail Test, if Exception_Status bit 7 = 1

- Request a Get_Attribute_Single, Exception_Detail_Alarm.
  Pass: Success_Response

Exception_Detail Structure Check

- For each non-zero Common_Except_Sz, verify each detail byte.
  Pass: Size = 2
  Pass: Common_Exception_Detail[0] bits 5 and 7 = 0.
  Pass: Common_Exception_Detail[1] bits 1 and 7 = 0.

Profile Specific Alarm/Warning Checks

- Verify the message data length is >= 5 < 8 for MFC.
- Verify the message data length is >= (4 + (2 * number of gauges)) for VPG, Alarm _Detail
- Verify the message data length is >= (4 + (3 * number of gauges)) for VPG, Warning _Detail
- Verify the message data length is >= 9 < 12 for PCV.
- For each non-zero Device_Except_Sz, verify each detail byte.
  Pass: Size = 1 for MFC
  Pass: Device Alarm_Exception_Detail bits 0, 6 and 7 = 0.
  Pass: Device Warning_Exception_Detail bits 6 and 7 = 0.
  Pass: Size = 5 for PCV
  Pass: Device Alarm_Exception_Detail byte 4, bits 2 - 7 = 0.
  Pass: Device Warning_Exception_Detail byte 4, bits 2 - 7 = 0.
- For Mfg_Except_Sz.
  Pass: Mfg_Except_Sz = 0 or 1 for MFC, VPG, PCV
  Repeat step 6.2 to verify the Exception_Detail_Warning attribute Structure

6.3) Alarm_Enable and Warning_Enable Test

- Request a Set_Attribute_Single, Alarm_Enable = 0.
- Request a Set_Attribute_Single, Warning_Enable = 0.
- Start a test timer for 5 seconds. Wait for timer to expire
- Request a Get_Attribute_Single, Exception_Status.
  Pass: value = 0 or 0x80 (No Alerts or Warnings)
- If Exception_Detail_Alarm is implemented, request a Get_Attribute_Single.
  Pass: value = 0 (No Alerts)
- If Exception_Detail_Warning is implemented, request a Get_Attribute_Single.
  Pass: value = 0 (No Warnings)

6.4) Clock_Behavior and Time Test, If Time is implemented,
- Request an Identity Object Reset, type = 0.
- When the Network Access is complete, request a Get_Attribute_Single, Time.
  Pass: If Clock_Behavior = 0, New Time < pre-Reset Time from step 2.
  Pass: If Clock_Behavior = 1 or 2, New Time > pre-Reset Time from step 2.

6.5) Non-Volatile attributes Test.
- Request a Set_Attribute_Single for each settable attribute, non-default value.
- Request a Get_Attribute_Single for each implemented attribute.
- Request an Identity Object Reset, type = 0.
- When Network Access is complete, request Get_Attribute_Single for each implemented non-volatile attribute.
  Pass: Each value = pre-Reset value.

6.6) String Attributes Test
- Request a Get_Attribute_Single for each implemented SHORT_STRING attribute.
  Pass: Each attribute has correct length and printable characters.

6.7) State Transition Test
- Request a Get_Attribute_Single, Device_Status.
- If Device_Status = 2, Idle, skip to step 6.8.2.

6.7.1) State = 3, Self-Test-Exception
- Request Start service.
  Pass: Error_Response, 94 0C ff, Object_State_Conflict
- Request Abort service.
  Pass: Error_Response, 94 0C ff, Object_State_Conflict
- Request Stop service.
  Pass: Error_Response, 94 0C ff, Object_State_Conflict
- Request a Perform_Diagnostics service, type 0.
  Pass: Success_Response.
- Request a Recover service.
  Pass: Success_Response.
- Request an S-Device Supervisor Reset service.
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 2, Idle. Continue with step 6.8.2.
  Pass: Device_Status = 1, Self-Test.

Start a test timer for 2 seconds, when timer expires

- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 2, Idle. Continue with step 6.8.2
- Request Identity object Reset type 0.
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 3, Self-Test-Exception. Exit Step 6.8.

6.7.2) State = Idle

- Request an S-Device Supervisor Reset service.
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 2, Idle.
- Request a Stop service.
  Pass: Error_Response 94 0b ff, Already In Requested Mode/State.
- Request a Recover service.
  Pass: Error_Response 94 0c ff, Object State Conflict.
- Request a Perform_Diagnostics service, type 0.
  Pass: Success_Response.
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 1, Self-Test, or 2, Idle. If Self-Test, repeat request.

State Transition, Idle -> Executing

- Open an I/O connection, EPR = 100 ms, send I/O data.
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 4, Executing.

6.7.3) State = Executing

- Request a Start service.
  Pass: Error_Response 94 0b ff, Already In Requested Mode/State.
- Request a Recover service.
  Pass: Error_Response 94 0c ff, Object State Conflict.

State Transition, Executing -> Idle

- Start a test timer for 500 ms and allow it time out.
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 2, Idle.

State Transition, Idle -> Abort

- Request an Abort service.

6.7.4) State = Abort

- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 5, Abort.
- Request a Start service.
  Pass: Error_Response 94 0c ff, Object State Conflict.
- Request a Stop service.
  Pass: Error_Response 94 0c ff, Object State Conflict.
- Request an Abort service.

Pass: Error_Response 94 0b ff, Already In Requested Mode/State.

- Request a Perform_Diagnostics service, type 0.
  Pass: Success_Response.
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 5, Abort.
- Open an I/O connection, EPR = 50 ms and allow it time out
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 5, Abort.

State Transition, Abort -> Idle

- Request a Recover service.
  Pass: Success_Response. Transition to Idle
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 2, Idle.
- Open an I/O connection, EPR = 50 ms and allow it time out
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 2, Idle.

State Transition, Idle -> Abort

- Request an Abort service.
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 5, Abort.

State Transition, Abort-> Self-Testing

- Request an S-Device Supervisor Reset service.
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 1, Self-Test, or 2, Idle. If Self-Test, repeat request.

State Transition, Idle -> Executing

- Request a Start service.
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 4, Executing.

6.7.5) State = Executing

State Transition, Executing -> Idle

- If Stop is supported, Request a Stop service.
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 2, Idle.
- Request a Start service.
- Request a Perform_Diagnostics service.
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 1, Self-Test, or 2, Idle. If Self-Test, repeat request.

State Transition, Executing -> Abort

- Request an Abort service.
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 5, Abort.

State Transition, Abort -> Idle -> Executing

- Request a Recover service.
- Request a Start service.
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 4, Executing.
- Request an S-Device Supervisor Reset service.
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 1, Self-Test, or 2, Idle. If Self-Test, repeat request.
- Request a Start service.
- Open an I/O connection, EPR = 50 ms and Release it
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 2, Idle.

6.8) Identity Reset Interaction Test

- Request a Start service.
- Request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 4, Executing.
- Request an Identity object Reset service.
- When the Network Access is complete, request a Get_Attribute_Single, Device_Status.
  Pass: Device_Status = 1, Self-Test, or 2, Idle. If Self-Test, repeat request.

6.9) Profile Specific Checks

MFC Profile

- Verify that the Device Type Attribute value is “MFC” or “MFM”.
  Pass: “MFC” for Mass Flow Controller.
  Pass: “MFM” for Mass Flow Meter.

VPG Profile

- Verify that the Device Type Attribute value is “VG” or “CG”, “MG”.
  Pass: “VG” for Vacuum Gauge.
  Pass: “CG” for Combination Guage.
  Pass: “MG” for Multiple Guage.

PCV Profile

- Verify that the Device Type Attribute value is “PCV”.
  Pass: “PCV” for Process Control Valve.

RGA Profile

- Verify that the Device Type Attribute value is “RGA”.
  Pass: “RGA” for Process Control Valve.

DCG Profile

- Verify that the Device Type Attribute value is “DCG”.
  Pass: “DCG” for Process Control Valve.

RFG Profile

- Verify that the Device Type Attribute value is “RFG”.
  Pass: “RFG” for Process Control Valve.

Turbo Pump Profile

- Verify that the Device Type Attribute value is “Turbo Pump”.
  Pass: “Turbo Pump” for Process Control Valve.

6.10) If Instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.34 S-Analog Sensor Object Test x31

This section defines the conformance test for the S-Analog Sensor object. This test is required when the S-Analog Sensor object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the S-Analog Sensor object.

2) Instance attributes access rules for the S-Analog Sensor object.

3) Common Service implementations for class and instance.

4) Object–specific and Reserved service implementations for class and instance.

5) Conformance when incorrect data is used to access attributes and services.

5.1) response when Set_Attribute_Single is requested with invalid data.

6) Object behavior accessible from the network.

6.1) Attribute Access when S-Analog Sensor is in Executing state.

6.2) Data_Type configuration for various Assembly Instances.

6.3) Value and Safe_State/Safe_Value behavior.

6.4) Zero_Adjust and Gain_Adjust services implementation.

6.5) non-volatile attributes behavior.

6.6) implementation of the Attributes_List attribute.

Procedure Definition

- Initialization
  The S-Analog Sensor object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection, EPR = 2500 ms.
- Request a Get_Attribute_Single, S-Device Supervisor, Device_Status.
- If Device_Status does not = Idle, request an S-Device Supervisor Reset service.
- Request a Get_Attribute_Single, S-Device Supervisor, Device_Status.
  Save the Device_Status.

1) Class attributes access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(6..199), Service_Not_Supported, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Max_Instance_Attribute_Id (07)	[UINT(7..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..31)	[Service_Not_Supported, Attribute_Not_Supported]
Class_Level_Status_Extensioin (32)	[USINT(0,1), Service_Not_Supported, Attribute_Not_Supported]
Undefined (33..98)	[Service_Not_Supported, Attribute_Not_Supported] ***Note C1***
Subclass (99)	[UINT (0,1), Service_Not_Supported, Attribute_Not_Supported] ***Note C1***

Note C1: Values for these attributes are defined and limited by the Device Profile

Vacuum/Pressure Gage (1B)	Subclass = 1, Instance Selector
Attribute Name (ID)	**[Expected Values, Responses]**
Active_Value (94)	[INT, or defined by Data_Type, Attribute_Not_Supported]
Active_Instance_Number (95)	[UINT(1), Attribute_Not_Supported]
Number_of_Gauges (96)	[USINT, Attribute_Not_Supported]
Subclass (99)	(1)

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attributes access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[USINT, Attribute_Not_Supported] ***Note 1***
Attributes_List (02)	[USINT[Number_of_Attributes], Attribute_Not_Supported]
Data_Type (03)	[USINT, Attribute_Not_Supported] ***Note 1***
Data_Units (04)	[UINT, Attribute_Not_Supported] ***Note 1***
Reading_Valid (05)	[BOOL]
Value (06)	[INT or defined by Data_Type]
Status (07)	[BYTE(0x0..0x0F)]
Alarm_Enable (08)	[BOOL(0), Attribute_Not_Supported]
Warning_Enable (09)	[BOOL(0), Attribute_Not_Supported]
Full_Scale (10)	[INT(65535) defined by Data_Type, Attribute_Not_Supported]
Offset_A_Data_Type (11)	[USINT, Attribute_Not_Supported] **Note 1**

Attribute Name (ID)	[Expected Values, Responses]
Offset_A (12)	[INT(0), or defined by Data_Type, Attribute_Not_Supported]
Gain_Data_Type (13)	[USINT, Attribute_Not_Supported] **Note 1**
Gain (14)	[REAL(1.0), or defined by Gain_Data_Type, Attribute_Not_Supported]
Unity_Gain_Reference (15)	[REAL(1.0), or defined by Gain_Data_Type, Attribute_Not_Supported]
Offset_B (16)	[INT(0), or defined by Data_Type, Attribute_Not_Supported]
Alarm_Trip_Point_High (17)	[INT(max) or defined by Data_Type, Attribute_Not_Supported]
Alarm_Trip_Point_Low (18)	[INT(min) or defined by Data_Type, Attribute_Not_Supported]
Alarm_Hysteresis (19)	[INT(0), or defined by Data_Type, Attribute_Not_Supported]
Alarm_Settling_Time (20)	[UINT(0), Attribute_Not_Supported]
Warning_Trip_Point_High (21)	[INT(max) or defined by Data_Type, Attribute_Not_Supported]
Warning_Trip_Point_Low (22)	[INT(min) or defined by Data_Type, Attribute_Not_Supported]
Warning_Hysteresis (23)	[INT(0) or defined by Data_Type, Attribute_Not_Supported]
Warning_Settling_Time (24)	[UINT(0), Attribute_Not_Supported]
Safe_State (25)	[USINT(0), Attribute_Not_Supported]
Safe_Value (26)	[INT(0) or defined by Data_Type, Attribute_Not_Supported]
AutoZero_Enable (27)	[BOOL(0), Attribute_Not_Supported]
AutoZero_Status (28)	[BOOL(0), Attribute_Not_Supported]
AutoRange_Enable (29)	[BOOL(0), Attribute_Not_Supported]
Range_Multiplier (30)	[REAL(1.0), Attribute_Not_Supported]
Averaging_Time (31)	[UINT(0), Attribute_Not_Supported]
Overrange (32)	[INT(max) or defined by Data_Type, Attribute_Not_Supported]
Underrange (33)	[INT(min) or defined by Data_Type, Attribute_Not_Supported]
Produce_Trigger_Delta (34)	[INT(0), Attribute_Not_Supported]
Gas_Calibration_Instance (35)	[UINT(0), Attribute_Not_Supported]
Produce_Trigger_Delta_Type (36)	[USINT(0,1), Attribute_Not_Supported] **Note 1**
Value_Descriptor (37)	[SHORT_STRING (Max 20 chars), Attribute_Not_Supported
Undefined (38..98)	[Attribute_Not_Supported] **Note 2**
Subclass (99)	[UINT(0..5)] **Note 1**

Note 1: Values for these attributes are defined and limited by the Device Profile
Note 2: These attributes are defined and limited by the Device Profile

Mass Flow Controller (1A)	
Attribute Name (ID)	**[Expected Values, Responses]**
Number_of_Attributes (01)	(3..38)
Data_Type (03)	(0xC3 (default) , 0xCA)
Data_Units (04)	(0x1001(default), 0x1400..0x1410)
Offset_A_Data_Type (11)	(0xC3(default), 0xCA)
Gain_Data_Type (13)	(0xC3, 0xCA(default))
Produce_Trigger_Delta (34)	[INT(0), or defined by Data_Type, Attribute_Not_Supported]
Subclass (99)	(0, 1)

Vacuum/Pressure Gauge (1C)	
Attribute Name (ID)	**[Expected Values, Responses]**
Number_of_Attributes (01)	(3..53) depending on Subclass value
Data_Type (03)	(0xC3 (default) , 0xCA)
Data_Units (04)	(0x1001(default), 0x1300..0x130C)
Offset_A_Data_Type (11)	(0xC3(default), 0xCA)
Gain_Data_Type (13)	(0xC3, 0xCA(default))
Produce_Trigger_Delta_Type (36)	[USIINT(1)]
Subclass (99)	(0, 2..5)

Process Control Valve (1D)	
Attribute Name (ID)	**[Expected Values, Responses]**
Number_of_Attributes (01)	(3..37)
Data_Type (03)	(0xC3 (default) , 0xCA)
Data_Units (04) instance 1, 2	(0x1001(default), 0x1007, 0x1301, 0x1302, 0x1307..0x130A)
Data_Units (04) instance 3	(0, 0x1001(default), 0x1007, 0x1703)
Data_Units (04) instance 4	(0, 0x1001(default), 0x1007, 0x1400, 0x1401)
Data_Units (04) instance 5	(0, 0x1001(default), 0x1200, 0x1201)
Data_Units (04) instance 6..10	(0, 0x1001(default), 0x1007, 0x2D00)
Offset_A_Data_Type (11)	(0xC3(default), 0xCA)
Gain_Data_Type (13)	(0xC3(default), 0xCA)
Subclass (99)	(0, 6)

Mass Flow Controller (1A)	**SubClass = 1, Flow Diagnostics**
Attribute Name (ID)	**[Expected Values]**
Flow_Totalizer (95)	[ULINT(0), Attribute_Not_Supported]
Flow_Hours (96)	[UDINT(0), Attribute_Not_Supported]

Vacuum/Pressure Gauge (1C)	**SubClass = 2, Heat Transfer Vacuum Gauge**
Attribute Name (ID)	**[Expected Values]**
Cable_Length (91)	[UINT(0), Attribute_Not_Supported]
High_Temp_Warning_Value (92)	[REAL(0), Attribute_Not_Supported]
Sensor_Temperature (93)	[REAL, Attribute_Not_Supported]
Sensor_Warning (94)	[(BYTE(0), BYTE(0)), Attribute_Not_Supported]
Sensor_Alarm (95)	[(BYTE(0), BYTE(0)), Attribute_Not_Supported]
Status_Extension (96)	[BYTE(0), Attribute_Not_Supported]

Vacuum/Pressure Gauge (1C)	**SubClass = 3, Diaphram Gauge**
Attribute Name (ID)	**[Expected Values]**
Sensor_Temperature (93)	[REAL, Attribute_Not_Supported]
Sensor_Warning (94)	[(BYTE(0), BYTE(0)), Attribute_Not_Supported]
Sensor_Alarm (95)	[(BYTE(0), BYTE(0)), Attribute_Not_Supported]

Vacuum/Pressure Gauge (1C)	**SubClass = 3, Diaphram Gauge**
Attribute Name (ID)	**[Expected Values]**
Status_Extension (96)	[BYTE(0), Attribute_Not_Supported]

Vacuum/Pressure Gauge (1C)	**SubClass = 4, Cold Cathode Ion Gauge**
Attribute Name (ID)	**[Expected Values]**
High_Voltage (91)	[REAL, Attribute_Not_Supported]
Sensitivity (92)	[REAL(1.0), Attribute_Not_Supported]
High_Voltage_Status (93)	[BOOL]
Sensor_Warning (94)	[(BYTE(0), BYTE(0)), Attribute_Not_Supported]
Sensor_Alarm (95)	[(BYTE(0), BYTE(0)), Attribute_Not_Supported]
Status_Extension (96)	[BYTE(0), Attribute_Not_Supported]

Vacuum/Pressure Gauge (1C)	**SubClass = 5, Hot Cathode Ion Gauge**
Attribute Name (ID)	**[Expected Values]**
Filament_Bias_Voltage (79)	[REAL, Attribute_Not_Supported]
Grid_Voltage (80)	[REAL, Attribute_Not_Supported]
Active_Degas_Filament (81)	[BYTE, Attribute_Not_Supported]
Degas_Power (82)	[REAL, Attribute_Not_Supported]
Filamnent_Current (83)	[REAL, Attribute_Not_Supported]
Degas_TimeOff_Remaining (84)	[UINT(0), Attribute_Not_Supported]
Degas_TimeOff_Interval (85)	[UINT, Attribute_Not_Supported]
Degas_TimeOn_Remaining (86)	[UINT(0), Attribute_Not_Supported]
Degas_TimeOff_Interval (87)	[UINT, Attribute_Not_Supported]
Degas_Status (88)	[BOOL, Attribute_Not_Supported]
Active _Filament (89)	[BYTE, Attribute_Not_Supported]
Sensitivity (90)	[REAL(1.0), Attribute_Not_Supported]
Emmsion_Current (91)	[REAL, Attribute_Not_Supported]
Mode _Filament_Selection (92)	[BOOL(1), Attribute_Not_Supported]
Emmision_Status (93)	[BOOL]
Sensor_Warning (94)	[(BYTE(0), BYTE(0)), Attribute_Not_Supported]
Sensor_Alarm (95)	[(BYTE(0), BYTE(0)), Attribute_Not_Supported]
Status_Extension (96)	[BYTE(0), Attribute_Not_Supported]

Process Control Valve (1D)	**SubClass = 6, Transfer Function**
Attribute Name (ID)	**[Expected Values]**
Pressure_Variable_Map_Function(96)	[USINT(0), Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	**[Expected Responses]**
Undefined (00)	[Attribute_Not_Supported (x14)]
Num_Attributes (01)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes_List (02)	[Attribute_Not_Settable, Attribute_Not_Supported]
Data_Type (03)	[Success_Response, Attribute_Not_Supported]
Data_Units (04)	[Success_Response, Attribute_Not_Supported]
Reading_Valid (05)	[Attribute_Not_Settable]
Value (06)	[Attribute_Not_Settable]
Status (07)	[Attribute_Not_Settable]
Alarm_Enable (08)	[Success_Response, Attribute_Not_Supported]
Warning_Enable (09)	[Success_Response, Attribute_Not_Supported]
Full_Scale (10)	[Attribute_Not_Settable, Attribute_Not_Supported]
Offset_A_Data_Type (11)	[Attribute_Not_Settable, Attribute_Not_Supported]
Offset_A (12)	[Success_Response, Attribute_Not_Supported]
Gain_Data_Type (13)	[Attribute_Not_Settable, Attribute_Not_Supported]
Gain (14)	[Success_Response, Attribute_Not_Supported]
Unity_Gain_Reference (15)	[Attribute_Not_Settable, Attribute_Not_Supported]
Offset_B (16)	[Success_Response, Attribute_Not_Supported]
Alarm_Trip_Point_High (17)	[Success_Response, Attribute_Not_Supported]
Alarm_Trip_Point_Low (18)	[Success_Response, Attribute_Not_Supported]
Alarm_Hysteresis (19)	[Success_Response, Attribute_Not_Supported]
Alarm_Settling_Time (20)	[Success_Response, Attribute_Not_Supported]
Warning_Trip_Point_High (21)	[Success_Response, Attribute_Not_Supported]
Warning_Trip_Point_Low (22)	[Success_Response, Attribute_Not_Supported]
Warning_Hysteresis (23)	[Success_Response, Attribute_Not_Supported]
Warning_Settling_Time (24)	[Success_Response, Attribute_Not_Supported]
Safe_State (25)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Safe_Value (26)	[Success_Response, Attribute_Not_Supported]
AutoZero_Enable (27)	[Success_Response, Attribute_Not_Supported]
AutoZero_Status (28)	[Attribute_Not_Settable, Attribute_Not_Supported]
AutoRange_Enable (29)	[Success_Response, Attribute_Not_Supported]
Range_Multiplier (30)	[Attribute_Not_Settable, Attribute_Not_Supported]
Averaging_Time (31)	[Success_Response, Attribute_Not_Supported]
Overrange (32)	[Attribute_Not_Settable, Attribute_Not_Supported]
Underrange (33)	[Attribute_Not_Settable, Attribute_Not_Supported]
Produce_Trigger_Delta (34)	[Success_Response, Attribute_Not_Supported]
Gas_Calibration_Instance (35)	[Success_Response, Attribute_Not_Supported]
Produce_Trigger_Delta (36)	[Success_Response, Attribute_Not_Supported]
Value_Descriptor (37)	[Success_Response, Attribute_Not_Supported
Undefined (38..98)	[Attribute_Not_Supported] **Note 3**

Attribute Name (ID)	[Expected Responses]
Subclass (99)	[Attribute_Not_Settable, Attribute_Not_Supported]

Note 3: Values for these attributes are defined and limited by the Device Profile

Mass Flow Controller (1A)	Suubclass =1, Flow Diagnostics
Attribute Name (ID)	**[Expected Values]**
Flow_Totalizer (95)	[Success_Response, Attribute_Not_Supported]
Flow_Hours (96)	[Success_Response, Attribute_Not_Supported]

Vacuum/Pressure Gage (1C)	SubClass = 2, Heat Transfer Vacuum Gage
Attribute Name (ID)	**[Expected Values]**
Cable_Length (91)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
High_Temp_Warning_Value (92)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Sensor_Temperature (93)	[Attribute_Not_Settable, Attribute_Not_Supported]
Sensor_Warning (94)	[Attribute_Not_Settable, Attribute_Not_Supported]
Sensor_Alarm (95)	[Attribute_Not_Settable, Attribute_Not_Supported]
Status_Extension (96)	[Attribute_Not_Settable, Attribute_Not_Supported]

Vacuum/Pressure Gage (1C)	SubClass = 3, Diaphram Gage
Attribute Name (ID)	**[Expected Values]**
Sensor_Temperature (93)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Sensor_Warning (94)	[Attribute_Not_Settable, Attribute_Not_Supported]
Sensor_Alarm (95)	[Attribute_Not_Settable, Attribute_Not_Supported]
Status_Extension (96)	[Attribute_Not_Settable, Attribute_Not_Supported]

Vacuum/Pressure Gage (1C)	SubClass = 4, Cold Cathode Ion Gage
Attribute Name (ID)	**[Expected Values]**
High_Voltage (91)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Sensitivity (92)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
High_Voltage_Status (93)	[Attribute_Not_Settable]
Sensor_Warning (94)	[Attribute_Not_Settable, Attribute_Not_Supported]
Sensor_Alarm (95)	[Attribute_Not_Settable, Attribute_Not_Supported]
Status_Extension (96)	[Attribute_Not_Settable, Attribute_Not_Supported]

Vacuum/Pressure Gauge (1C)	SubClass = 5, Hot Cathode Ion Gauge
Attribute Name (ID)	**[Expected Values]**
Filament_Bias_Voltage (79)	[Attribute_Not_Settable, Attribute_Not_Supported]
Grid_Voltage (80)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Active_Degas_Filament (81)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Degas_Power (82)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Filamnent_Current (83)	[Attribute_Not_Settable, Attribute_Not_Supported]
Degas_TimeOff_Remaining (84)	[Attribute_Not_Settable, Attribute_Not_Supported]

Vacuum/Pressure Gauge (1C)	SubClass = 5, Hot Cathode Ion Gauge
Attribute Name (ID)	**[Expected Values]**
Degas_TimeOff_Interval (85)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Degas_TimeOn_Remaining (86)	[Attribute_Not_Settable, Attribute_Not_Supported]
Degas_TimeOff_Interval (87)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Degas_Status (88)	[Attribute_Not_Settable, Attribute_Not_Supported]
Active _Filament (89)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Sensitivity (90)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Emmsion_Current (91)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Mode _Filament_Selection (92)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Emmision_Status (93)	[Attribute_Not_Settable]
Sensor_Warning (94)	[Attribute_Not_Settable, Attribute_Not_Supported]
Sensor_Alarm (95)	[Attribute_Not_Settable, Attribute_Not_Supported]
Status_Extension (96)	[Attribute_Not_Settable, Attribute_Not_Supported]

Process Control Valve (1D)	SubClass = 6, Transfer Function
Attribute Name (ID)	**[Expected Values]**
Pressure_Variable_Map_Function(96)	[Success_Response, Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response, Service_Not_Supported (if no attributes are supported)]
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (code)	[Expected Responses]
Reserved (00..13)	[Service_Not_Supported (x08)]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.
- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Zero_Adjust (75)	[Success_Response, Service_Not_Supported (x08)]
Gain_Adjust (76)	[Success_Response, Service_Not_Supported]
Object–specific Codes (77..99)	[Service_Not_Supported] Note 4

Note 4: These subclass specific services are defined and limited by the Device Profile

Vacuum/Pressure Gauge (1C)	SubClass = 4, Cold Cathode Ion Gauge
Service Name (Code)	**[Expected Responses]**
Set_High_Voltage_State (98)	[Success_Response, Object_State_Conflict]
Clear_High_Voltage_Off_Alarm(99)	[Success_Response, Object_State_Conflict]

Vacuum/Pressure Gauge (1C)	SubClass = 5, Hot Cathode Ion Gauge
Service Name (Code)	**[Expected Responses]**
Set_Degas_State (97)	[Success_Response, Service_Not_Supported]
Set_Emission_State (98)	[Success_Response]
Clear_Emission _Off_Alarm (99)	[Success_Response]

- Request each Reserved Service, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..104)	[Success_Response , Service_Not_Supported] Profile Specific
Reserved Codes (105..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) For each attribute that is implemented with Set access,

- Request a Set_Attribute_Single service, Alarm_Enable = 254.
- Request a Set_Attribute_Single service, Warning_Enable = 254.
- Request a Set_Attribute_Single service, AutoZero_Enable = 254.
- Request a Set_Attribute_Single service, AutoRange_Enable = 254.
- Request a Set_Attribute_Single service, Safe_State = 4..50.
- Request a Set_Attribute_Single service, Data_Type = (0x00..0xFF, except 0xC3, 0xCA).
- Request a Set_Attribute_Single service, Data_Units = Invalid Values, see Note 1

Pass: For each attribute, Error_Response 94 09 FF, Invalid_Attribute_Value.

- Request an Zero_Adjust service with parameter = 5 bytes.
  Pass: Error_Response, 94 15 ff, Too_Much_Data.
- Request an Gain_Adjust service with parameter = 5 bytes.
  Pass: Error_Response, 94 15 ff, Too_Much_Data.
- Request an Set_High_Voltage_State service with no parameter.
  Pass: Error_Response, 94 13 ff, Not_Enough_Data.
- Request an Set_High_Voltage_State service with parameter = 2 bytes.
  Pass: Error_Response, 94 15 ff, Too_Much_Data.
- Request an Set_High_Voltage_State service with parameter = 0x07.
  Pass: Error_Response, 94 20 ff, Invalid_Parameter.

6) Behavior tests.
6.1) Atribute Access S-Analog Sensor state = Executing.

- Request an S-Device Supervisor Start service.
- Request a Get_Attribute_Single, S-Device Supervisor, Device_Status.
  If Device_Status does not = 4, Executing, skip to step 6.2.
- Request a Set_Attribute_Single, Data_Type.
  Pass: Error_Response, 94 0C ff, Object State Conflict.
- Request a Set_Attribute_Single, Gain_Data_Type.
  Pass: Error_Response, 94 0C ff, Object State Conflict.

6.2) Data_Type configuration for Assembly Instances, MFC, V/PG, PCV, and FFC.

1 Assembly Instances for S-Analog Sensor Test 6.2

Profile	Integer Input	Integer Output	Floating Point Input	Floating Point Output
MFC	6	7	18	19
V/PG	2		4 (if implemented)	
PCV	2	18	7	23
FFC	1	10	2	11

- Open an I/O connection
- Configure the produced and consumed connection path to integer data type assembly.
- Request a Set_Attribute_Single, I/O EPR = 1000ms.
- Verify the Produced and Consumed Connection Sizes.
  Pass: Produced and Consumed Sizes = Connection Defaults for integer type assembly.
- Request a Set_Attribute_Single, Data_Type.
  Pass: Error_Response, 94 0C ff, Object State Conflict.
- Close/Release the I/O connection.
- Open an I/O connection.
- Configure the produced and consumed connection path to floating type assembly instance.
- Verify the Produced and Consumed Connection Sizes.
  Pass: Produced and Consumed Sizes = Floating Point Defaults.

- Request a Set_Attribute_Single, I/O EPR = 1000ms.
- Request a Get_Attribute_Single, Data_Type.
  Pass: Success_Response, 0xCA (REAL).
- Close/Release the I/O connection.
- Reset the device (Identity Reset type 1).
- Open an I/O connection.
- Verify the Produced and Consumed Connection Sizes.
  Pass: Produced and Consumed Sizes = Floating Point Defaults.
- Configure the produced and consumed connection path to integer data type assembly instance.
- Close/Release the I/O connection.

6.3) Value and Safe_State/Safe_Value behavior.
Only implemented attributes and supported values are tested

- Request an S-Device Supervisor Start service.
- Request an S-Device Supervisor Abort service.
- Request a Get_Attribute_Single, Value.
  Pass: Value = 0.
- Request an S-Device Supervisor Recover service.
- Request an S-Device Supervisor Start service.
- Request a Set_Attribute_Single, Safe_State = 1.
- Request an S-Device Supervisor Abort service.
- Request a Get_Attribute_Single, Value.
  Pass: Value = Full Scale.
- Request an S-Device Supervisor Recover service.
- Request an S-Device Supervisor Start service.
- Request a Set_Attribute_Single, Safe_State = 2.
- Request an S-Device Supervisor Abort service.
- Request a Get_Attribute_Single, Value.
  Pass: Value = Full Scale, or 0 if Safe_State = 1 not supported.
- Request an S-Device Supervisor Recover service.
- Request an S-Device Supervisor Start service.
- Request a Set_Attribute_Single, Safe_State = 3.
- Request a Get_Attribute_Single, Safe_Value.
- Request an S-Device Supervisor Abort service.
- Request a Get_Attribute_Single, Value.
  Pass: Value = Safe_Value, default of 0.
- Request a Set_Attribute_Single, Safe_Value = 1.
- Request an S-Device Supervisor Recover service.
- Request an S-Device Supervisor Start service.
- Request an S-Device Supervisor Abort service.
- Request a Get_Attribute_Single, Value.
  Pass: Value = Safe_Value = 1.
- Request an S-Device Supervisor Recover service.

6.4) Zero_Adjust and Gain_Adjust services implementation.

Only implemented services are tested

To Be Completed

- Request an Zero_Adjust service with no parameter.
  Pass: Success_Response.
- Request an Zero_Adjust service with parameter = 0.
  Pass: Success_Response.
- Request an Gain_Adjust service with parameter = 0.
  Pass: Success_Response.

6.5) Non-Volatile attributes Test.

- Request a Set_Attribute_Single for each settable attribute, non-default value.
- Request a Get_Attribute_Single for each implemented attribute.
- Request an Identity Object Reset, type = 0.
- When the Network Access is complete, request a Get_Attribute_Single for each implemented non-volatile attribute.
  Pass: Each value = pre-Reset value.

6.6) If Instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.35 S-Analog Actuator Object Test x32

This section defines the conformance test for the S-Analog Actuator object. This test is required when the S-Analog Actuator object is implemented in the device.

Functional Description

This test verifies the:
1) Class attributes access rules for the S-Analog Actuator object.
2) Instance attributes access rules for the S-Analog Actuator object.
3) Common Service implementations for class and instance.
4) Object–specific and Reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
5.1) response when Set_Attribute_Single is requested with invalid data.
6) Object behavior accessible from the network.
6.1) Attribute Access when S-Analog Actuator is in Executing state.
6.2) Data_Type configuration for various Assembly Instances.
6.3) Alarm and Warning Trip Point and Hysteresis behavior.
6.4) Value and Safe_State/Safe_Value behavior.
6.5) non-volatile attributes behavior.
6.6) implementation of the Attributes_List attribute.

Procedure Definition

- Initialization
  The S- Analog Actuator object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection, EPR = 2500 ms.
- Request a Get_Attribute_Single, S-Device Supervisor, Device_Status.
- If Device_Status does not = Idle, request an S-Device Supervisor Reset service.
  Request a Get_Attribute_Single, S-Device Supervisor, Device_Status. Save the Device_Status.

1) Class attributes access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(6..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(7..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..98)	[Service_Not_Supported, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Subclass (99)	[UINT (0), Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attributes access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[USINT, Attribute_Not_Supported] **Note 1**
Attributes_List (02)	[USINT[Number_of_Attributes], Attribute_Not_Supported]
Data_Type (03)	[USINT, Attribute_Not_Supported] **Note 1**
Data_Units (04)	[UINT, Attribute_Not_Supported] **Note 1**
Override (05)	[USINT(0..4, 64..255)]
Value (06)	[INT or defined by Data_Type]
Status (07)	[BYTE(0x0..0x0F)]
Alarm_Enable (08)	[BOOL(0), Attribute_Not_Supported]
Warning_Enable (09)	[BOOL(0), Attribute_Not_Supported]
Offset (10)	[INT or defined by Data_Type, Attribute_Not_Supported]
Bias (11)	[INT or defined by Data_Type, Attribute_Not_Supported]
Gain_Data_Type (12)	[USINT, Attribute_Not_Supported] **Note 1**
Gain (13)	[REAL(1.0), or defined by Gain_Data_Type, Attribute_Not_Supported]
Unity_Gain_Reference (14)	[REAL(1.0), or defined by Gain_Data_Type, Attribute_Not_Supported]
Alarm_Trip_Point_High (15)	[INT or defined by Data_Type, Attribute_Not_Supported]
Alarm_Trip_Point_Low (16)	[INT or defined by Data_Type, Attribute_Not_Supported]
Alarm_Hysteresis (17)	[UINT(0), Attribute_Not_Supported]
Warning_Trip_Point_High (18)	[INT or defined by Data_Type, Attribute_Not_Supported]
Warning_Trip_Point_Low (19)	[INT or defined by Data_Type, Attribute_Not_Supported]
Warning_Hysteresis (20)	[INT(0) or defined by Data_Type, Attribute_Not_Supported]
Safe_State (21)	[USINT(0), Attribute_Not_Supported]
Safe_Value (22)	[INT(0) or defined by Data_Type, Attribute_Not_Supported]
Undefined (23..98)	[Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Subclass (99)	Note 1

Note 1: Values for these attributes are defined and limited by the Device Profile

Mass Flow Controller (1A)	
Attribute Name (ID)	**[Expected Values]**
Number_of_Attributes (01)	(3..23)
Data_Type (03)	(0xC3 (default) , 0xCA)
Data_Units (04)	(0x1001(default), 0x1400..0x1410)
Gain_Data_Type (12)	(0xC3, 0xCA(default))
Subclass (99)	(0)

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Num_Attributes (01)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes_List (02)	[Attribute_Not_Settable, Attribute_Not_Supported]
Data_Type (03)	[Success_Response, Attribute_Not_Supported]
Data_Units (04)	[Success_Response, Attribute_Not_Supported]
Override (05)	[Success_Response, Object_State_Conflict]
Value (06)	[Success_Response, Attribute_Not_Settable]
Status (07)	[Attribute_Not_Settable]
Alarm_Enable (08)	[Success_Response, Attribute_Not_Supported]
Warning_Enable (09)	[Success_Response, Attribute_Not_Supported]
Offset (10)	[Success_Response, Attribute_Not_Supported]
Bias (11)	[Success_Response, Attribute_Not_Supported]
Gain_Data_Type (12)	[Attribute_Not_Settable, Attribute_Not_Supported]
Gain (13)	[Success_Response, Attribute_Not_Supported]
Unity_Gain_Reference (14)	[Attribute_Not_Settable, Attribute_Not_Supported]
Alarm_Trip_Point_High (15)	[Success_Response, Attribute_Not_Supported]
Alarm_Trip_Point_Low (16)	[Success_Response, Attribute_Not_Supported]
Alarm_Hysteresis (17)	[Success_Response, Attribute_Not_Supported]
Warning_Trip_Point_High (18)	[Success_Response, Attribute_Not_Supported]
Warning_Trip_Point_Low (19)	[Success_Response, Attribute_Not_Supported]
Warning_Hysteresis (20)	[Success_Response, Attribute_Not_Supported]
Safe_State (21)	[Success_Response, Attribute_Not_Supported]
Safe_Value (22)	[Success_Response, Attribute_Not_Supported]
Undefined (23..98)	[Attribute_Not_Supported]
Subclass (99)	[Attribute_Not_Settable, Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response, Service_Not_Supported (if no attributes are supported)]
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..104)	[Success_Response , Service_Not_Supported] Profile Specific

Service Name (Code)	[Expected Responses]
Reserved Codes (105..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) For each attribute that is implemented with Set access,

- Request a Set_Attribute_Single service, Alarm_Enable = 254.
- Request a Set_Attribute_Single service, Warning_Enable = 254.
- Request a Set_Attribute_Single service, Safe_State = 4..50.
- Request a Set_Attribute_Single service, Data_Type = (0x00..0xFF, except 0xC3, 0xCA).
- Request a Set_Attribute_Single service, Data_Units = Invalid Values, see Note 1
  Pass: For each attribute, Error_Response 94 09 FF, Invalid_Attribute_Value.

6) Behavior tests.

6.1) Attribute Access when S-Analog Sensor is in Executing state.

- Request an S-Device Supervisor Start service.
- Request a Get_Attribute_Single, S-Device Supervisor, Device_Status.
  If Device_Status does not = 4, Executing., skip to step 6.2.
- Request a Set_Attribute_Single, Data_Type.
  Pass: Error_Response, 94 0C ff, Object State Conflict.
- Request a Set_Attribute_Single, Gain_Data_Type.
  Pass: Error_Response, 94 0C ff, Object State Conflict.
- Request a Set_Attribute_Single, Value.
  Pass: Success_Response

6.2) Data_Type configuration for Assembly Instances, MFC, V/PG and PCV

MFC - input/output assembly instances, integer = 6/7, floating point = 18/19
V/PG - input assembly instances, integer = 2, floating point = 4 (if implemented)
PCV - input/output assembly instances, integer = 2/18, floating point =7/23

- Open an I/O connection
- Configure the produced and consumed connection path to integer data type assembly.
- Request a Set_Attribute_Single, I/O EPR = 1000ms.
- Verify the Produced and Consumed Connection Sizes.
  Pass: Produced and Consumed Sizes = Connection sizes of assembly 6 and 7.
- Request a Set_Attribute_Single, Data_Type.
  Pass: Error_Response, 94 0C ff, Object State Conflict.
- Close/Release the I/O connection.
- Open an I/O connection.
- Configure the produced and consumed connection path to floating type assembly instance.
- Verify the Produced and Consumed Connection Sizes.
  Pass: Produced and Consumed Sizes = Floating Point Defaults.
- Request a Set_Attribute_Single, I/O EPR = 1000ms.
- Request a Get_Attribute_Single, Data_Type.
  Pass: Success_Response, 0xCA (REAL).
- Close/Release the I/O connection.

- Reset the device (Identity Reset type 1).
- Open an I/O connection.
- Verify the Produced and Consumed Connection Sizes.
  Pass: Produced and Consumed Sizes = Floating Point Defaults.
- Configure the produced and consumed connection path to integer data type assembly.
- Close/Release the I/O connection.

6.3) Alarm and Warning Trip Point and Hysteresis behavior.
Only implemented attributes are tested

- Request an S-Device Supervisor Start service.
- Request a Set_Attribute_Single, Alarm_Enable = 1.
- Request a Set_Attribute_Single, Alarm_Trip_Point_High = max_int/2 value.
- Request a Set_Attribute_Single, Alarm_Trip_Point_Low = (2 * min-int) value.
- Request a Set_Attribute_Single, Alarm_Hysteresis = 200.
- Request a Set_Attribute_Single, Value = (Alarm_Trip_Point_High - 100) .
- Request a Get_Attribute_Single, Status.
  Pass: Success_Response, 0x00.
- Request a Set_Attribute_Single, Value = (Alarm_Trip_Point_High + Alarm_Hysteresis) .
- Request a Get_Attribute_Single, Status.
  Pass: Success_Response, 0x01 for Alarm, 0x04 for Warning.
- Request a Set_Attribute_Single, Value = 0 .
- If Settling_Time is implemented, request a Get_Attribute_Single, Status.
  Pass: Success_Response, 0x01 for Alarm, 0x04 for Warning.
- Request a Get_Attribute_Single, Status.
  Pass: Success_Response, 0x00.
- Request a Set_Attribute_Single, Value = (Alarm_Trip_Point_Low + 100) .
- Request a Get_Attribute_Single, Status.
  Pass: Success_Response, 0x00.
- Request a Set_Attribute_Single, Value = Alarm_Trip_Point_Low.
- Request a Get_Attribute_Single, Status.
  Pass: Success_Response, 0x02 for Alarm, 0x08 for Warning.
- Request a Set_Attribute_Single, Value = 0 .
- Request a Get_Attribute_Single, Status.
  Pass: Success_Response, 0x00 for Alarm, 0x04 for Warning.
  Repeat step 6.3 with the Warning Enable, Trip and Hysteresis Attributes.

6.4) Value and Safe_State/Safe_Value behavior.
Only implemented attributes are tested

- Request an S-Device Supervisor Start service.
- Request a Set_Attribute_Single, Value = 100 .
- Request an S-Device Supervisor Abort service.
- Request a Get_Attribute_Single, Value.
  Pass: Value = 0.
- Request an S-Device Supervisor Recover service.

- Request an S-Device Supervisor Start service.
- Request a Set_Attribute_Single, Safe_State = 1.
- Request an S-Device Supervisor Abort service.
- Request a Get_Attribute_Single, Value.
  Pass: Value = Full Scale.
- Request an S-Device Supervisor Recover service.
- Request an S-Device Supervisor Start service.
- Request a Set_Attribute_Single, Safe_State = 2.
- Request a Set_Attribute_Single, Value = 100 .
- Request an S-Device Supervisor Abort service.
- Request a Get_Attribute_Single, Value.
  Pass: Value = 100.
- Request an S-Device Supervisor Recover service.
- Request an S-Device Supervisor Start service.
- Request a Set_Attribute_Single, Safe_State = 3.
- Request an S-Device Supervisor Abort service.
- Request a Get_Attribute_Single, Value.
  Pass: Value = Safe_Value = 0 (default).
- Request an S-Device Supervisor Recover service.
- Request an S-Device Supervisor Start service.
- Request a Set_Attribute_Single, Safe_State = 3.
- Request a Set_Attribute_Single, Safe_Value = 1.
- Request an S-Device Supervisor Abort service.
- Request a Get_Attribute_Single, Value.
  Pass: Value = Safe_Value.
- Request an S-Device Supervisor Recover service.

6.5) Non-Volatile attributes Test.

- Request a Set_Attribute_Single for each settable attribute, non-default value.
- Request a Get_Attribute_Single for each implemented attribute.
- Request an Identity Object Reset, type = 0.
- When the Network Access is complete, request a Get_Attribute_Single for each implemented non-volatile attribute.
  Pass: Each value = pre-Reset value.

6.6) If Instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.36 S-Single Stage Controller Object Test x33

This section defines the conformance test for the S-Single Stage Controller object. This test is required when the S-Single Stage Controller object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the S-Single Stage Controller object.
2) Instance attributes access rules for the S-Single Stage Controller object.
3) Common Service implementations for class and instance.
4) Object–specific and Reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
5.1) response when Set_Attribute_Single is requested with invalid data.
5.2) response when implemented services requested with invalid data
6) Object behavior accessible from the network.
6.1) Attribute Access when S-Single Stage Controller is in Executing state.
6.2) Data_Type configuration for various Assembly Instances.
6.3) Control_Mode behavior.
6.4) Process_Variable and Control_Variable behavior.
6.5 Alarm and Warning behavior
6.6) Safe_State and Safe_Value behavior.
6.7) Ramp_Rate behavior.
6.8) PID & Source Select behavior.
6.9) Non-volatile attributes behavior.
6.10) Implementation of the Attributes_List attribute.

Procedure Definition

- Initialization
  The S-Single Stage Controller object must be available. Log the object name and test revision.
- Message Connection
  Establish an Explicit Messaging Connection, EPR = 2500 ms.
- Request a Get_Attribute_Single, S-Device Supervisor, Device_Status.
- If Device_Status does not = Idle, request an S-Device Supervisor Reset service.
  Request a Get_Attribute_Single, S-Device Supervisor, Device_Status. Save the Device_Status.

1) Class attributes access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Optional_Service_List (05)	[(UINT, UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(6..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(10..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..98)	[Service_Not_Supported, Attribute_Not_Supported]
Subclass (99)	[UINT (0), Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attributes access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[USINT, Attribute_Not_Supported] **Note 1**
Attributes_List (02)	[USINT[Number_of_Attributes], Attribute_Not_Supported]
Data_Type (03)	[USINT, Attribute_Not_Supported] **Note 1**
Data_Units (04)	[UINT, Attribute_Not_Supported] **Note 1**
Control_Mode (05)	[USINT(0..4, 128..255), Attribute_Not_Supported]
Setpoint(06)	[INT(0) or defined by Data_Type]
Process_Variable (07)	[INT(0) or defined by Data_Type] **Note 1**
CV_Data_Type (08)	[USINT, Attribute_Not_Supported] **Note 1**
Control_Variable (09)	[INT(0) or defined by CV_Data_Type] **Note 1**
Status (10)	[BYTE(0..3)]
Alarm_Enable (11)	[BOOL(0), Attribute_Not_Supported]
Warning_Enable (12)	[BOOL(0), Attribute_Not_Supported]
Alarm_Settling_Time (13)	[UINT(0), Attribute_Not_Supported]
Alarm_Error_Band (14)	[INT(0) or defined by Data_Type, Attribute_Not_Supported]
Warning_Settling_Time (15)	[UINT(0), Attribute_Not_Supported, Attribute_Not_Supported]
Warning_Error_Band (16)	[INT(0) or defined by Data_Type, Attribute_Not_Supported]
Safe_State (17)	[USINT(0), Attribute_Not_Supported]
Safe_Value (18)	[INT(0) or defined by Data_Type, Attribute_Not_Supported]
Ramp_Rate (19)	[UDINT(0), Attribute_Not_Supported]
Undefined (20..98)	[Attribute_Not_Supported] **Note 2**

Attribute Name (ID)	**[Expected Values, Responses]**
Subclass (99)	[UINT(0), Attribute_Not_Supported] **Note 1**

Note 1: Values for these attributes are defined and limited by the Device Profile

Note 2: Values for these attributes are defined and limited by the Device Profile

Mass Flow Controller (1A)	
Attribute Name (ID)	**[Expected Values, Responses]**
Number_of_Attributes (01)	(4..20)
Data_Type (03)	(0xC3 (default) , 0xCA)
Data_Units (04)	(0x1001(default), 0x1400..0x1410)
Process_Variable (07)	[Attribute_Not_Supported]
CV_Data_Type (08)	[Attribute_Not_Supported]
Control_Variable (09)	[Attribute_Not_Supported]
Subclass (99)	(0)

Process Control Valve (1D)	
Attribute Name (ID)	**[Expected Values]**
Number_of_Attributes (01)	(5..30)
Data_Type (03)	(0xC3 (default) , 0xCA)
Data_Units (04) instance 1	(0x1001(default), 0x1007, 0x1301, 0x1302, 0x1307..0x130A)
Data_Units (04) instance 2	(0, 0x1001(default), 0x1007, 0x1703)
Process_Variable (07)	[Success_Response]
Subclass (99)	(1)

Process Control Valve (1D)	**Subclass = 1, PID and Source Select**
Attribute Name (ID)	**[Expected Values, Responses]**
Calibrating_State (87)	[BOOL, Attribute_Not_Supported]
Delay (88)	[UINT, Attribute_Not_Supported]
Sensor_Crossover_High (89)	[INT or specified by Data_Type, Attribute_Not_Supported]
Sensor_Crossover_Low (90)	[INT or specified by Data_Type, Attribute_Not_Supported]
Expected_Process_Parameter (91)	[INT or specified by Data_Type, Attribute_Not_Supported]
Kd (92)	[REAL, Attribute_Not_Supported]
Ki (93)	[REAL, Attribute_Not_Supported]
Kp (94)	[REAL, Attribute_Not_Supported]
Control_Direction (95)	[BOOL, Attribute_Not_Supported]
Process_Variable_Source (96)	[USINT(0..5), Attribute_Not_Supported]
Subclass (99)	(1)

DC Power Generator (1F)	**Subclass = 2, DC Generator**
Attribute Name (ID)	**[Expected Values, Responses]**
Output_Max (85)	[DINT, Attribute_Not_Supported]
Output _Limit (86)	[DINT, Attribute_Not_Supported]

DC Power Generator (1F)	**Subclass = 2, DC Generator**
Attribute Name (ID)	**[Expected Values, Responses]**
Arc_Trip_Level (87)	[DINT, Attribute_Not_Supported]
Arc_Count_Limit (88)	[DINT(1), Attribute_Not_Supported]
Arc_Counter (89)	[UDINT, Attribute_Not_Supported]
Pulse_Enable (90)	[BOOL(0), Attribute_Not_Supported]
Pulse_Second_Setpoint (91)	[INT(0) or specified by Data_Type, Attribute_Not_Supported]
Pulse_Period_Duration (92)	[DINT, Attribute_Not_Supported]
Pulse_Duty_Cycle (93)	[UINT, Attribute_Not_Supported]
Sync_Enable (94)	[BOOL(0), Attribute_Not_Supported]
Sync_Phase_Delay (95)	[DINT, Attribute_Not_Supported]
Ramp_rate_Increment (96)	[UINT(0), Attribute_Not_Supported]
Subclass (99)	(2)

RF Power Generator (20 hex)	**Subclass = 3, RF Generator**
Attribute Name (ID)	**[Expected Values, Responses]**
Output_Max (88)	[DINT, Attribute_Not_Supported]
Output _Limit (89)	[DINT, Attribute_Not_Supported]
Pulse_Enable (90)	[BOOL(0), Attribute_Not_Supported]
Pulse_Second_Setpoint (91)	[INT(0) or specified by Data_Type, Attribute_Not_Supported]
Pulse_Period_Duration (92)	[DINT, Attribute_Not_Supported]
Pulse_Duty_Cycle (93)	[UINT, Attribute_Not_Supported]
Sync_Enable (94)	[BOOL(0), Attribute_Not_Supported]
Sync_Phase_Delay (95)	[DINT, Attribute_Not_Supported]
Ramp_rate_Increment (96)	[UINT(0), Attribute_Not_Supported]
Subclass (99)	(3)

RF Power Generator (20 hex)	**Subclass = 4, Frequency Control**
Attribute Name (ID)	**[Expected Values, Responses]**
Minimum_Frequency (92)	[DINT, Attribute_Not_Supported]
Maximum_Frequency (93)	[DINT, Attribute_Not_Supported]
Default_Frequency (94)	[DINT, Attribute_Not_Supported]
Frequency_Low_Limit (95)	[DINT, Attribute_Not_Supported]
Frequency_High_Limit (96)	[DINT, Attribute_Not_Supported]
Subclass (99)	(4)

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	**[Expected Responses]**
Undefined (00)	[Attribute_Not_Supported (x14)]
Num_Attributes (01)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported]

Attribute Name (ID)	[Expected Responses]
Attributes_List (02)	[Attribute_Not_Settable, Attribute_Not_Supported]
Data_Type (03)	[Success_Response, Attribute_Not_Supported]
Data_Units (04)	[Success_Response, Attribute_Not_Supported]
Control_Mode (05)	[Success_Response, Attribute_Not_Settable, Attribute_Not_Supported]
Setpoint (06)	[Success_Response]
Process_Variable (07)	Note 3
CV_Data_Type (08)	Note 3
Control_Varable (09)	Note 3
Status (10)	[Attribute_Not_Settable]
Alarm_Enable (11)	[Success_Response, Attribute_Not_Supported]
Warning_Enable (12)	[Success_Response, Attribute_Not_Supported]
Alarm_Settling_Time (13)	[Success_Response, Attribute_Not_Supported]
Alarm_Error_Band (14)	[Success_Response, Attribute_Not_Supported]
Warning_Settling_Time (15)	[Success_Response, Attribute_Not_Supported]
Warning_Error_Band (16)	[Success_Response, Attribute_Not_Supported]
Safe_State (17)	[Success_Response, Attribute_Not_Supported]
Safe_Value (18)	[Success_Response, Attribute_Not_Supported]
Ramp_Rate (19)	[Success_Response, Attribute_Not_Supported]
Undefined (20..98)	[Attribute_Not_Supported] Note 4
Subclass (99)	[Attribute_Not_Settable, Attribute_Not_Supported]

Note 3: Values for these attributes are defined and limited by the Device Profile
Note 4: Values for these attributes are defined and limited by the Device Profile

Mass Flow Controller (1A)	
Attribute Name (ID)	**[Expected Responses]**
Process_Variable (07)	[Attribute_Not_Supported]
CV_Data_Type (08)	[Attribute_Not_Supported]
Control_Variable (09)	[Attribute_Not_Supported]

Process Control Valve (1D)	Sunclass = 1, PID and Source Select
Attribute Name (ID)	**[Expected Values, Responses]**
Calibrating_State (87)	[Attribute_Not_Settable, Attribute_Not_Supported]
Delay (88)	[Success_Response, Attribute_Not_Supported]
Sensor_Crossover_High (89)	[Success_Response, Attribute_Not_Supported]
Sensor_Crossover_Low (90)	[Success_Response, Attribute_Not_Supported]
Expected_Process_Parameter (91)	[Success_Response, Attribute_Not_Supported]
Kd (92)	[Success_Response, Attribute_Not_Supported]
Ki (93)	[Success_Response, Attribute_Not_Supported]
Kp (94)	[Success_Response, Attribute_Not_Supported]
Control_Direction (95)	[Success_Response, Attribute_Not_Supported]

Process Control Valve (1D)	**Sunclass = 1, PID and Source Select**
Attribute Name (ID)	**[Expected Values, Responses]**
Process_Variable_Source (96)	[Success_Response, Attribute_Not_Supported]
Subclass (99)	[Attribute_Not_Settable]

DC Power Generator (1F)	**Subclass = 2, DC Generator**
Attribute Name (ID)	**[Expected Values, Responses]**
Output_Max (85)	[Attribute_Not_Settable, Attribute_Not_Supported]
Output _Limit (86)	[Success_Response, Attribute_Not_Supported]
Arc_Trip_Level (87)	[Success_Response, Attribute_Not_Supported]
Arc_Count_Limit (88)	[Success_Response, Attribute_Not_Supported]
Arc_Counter (89)	[Attribute_Not_Settable, Attribute_Not_Supported]
Pulse_Enable (90)	[Success_Response, Attribute_Not_Supported]
Pulse_Second_Setpoint (91)	[Success_Response, Attribute_Not_Supported]
Pulse_Period_Duration (92)	[Success_Response, Attribute_Not_Supported]
Pulse_Duty_Cycle (93)	[Success_Response, Attribute_Not_Supported]
Sync_Enable (94)	[Success_Response , Attribute_Not_Supported]
Sync_Phase_Delay (95)	[Success_Response, Attribute_Not_Supported]
Ramp_rate_Increment (96)	[Success_Response, Attribute_Not_Supported]
Subclass (99)	[Attribute_Not_Settable]

DC Power Generator (1F)	**Subclass = 3, RF Generator**
Attribute Name (ID)	**[Expected Values, Responses]**
Output_Max (88)	[Attribute_Not_Settable, Attribute_Not_Supported]
Output _Limit (89)	[Success_Response, Attribute_Not_Supported]
Pulse_Enable (90)	[Success_Response, Attribute_Not_Supported]
Pulse_Second_Setpoint (91)	[Success_Response, Attribute_Not_Supported]
Pulse_Period_Duration (92)	[Success_Response, Attribute_Not_Supported]
Pulse_Duty_Cycle (93)	[Success_Response, Attribute_Not_Supported]
Sync_Enable (94)	[Success_Response , Attribute_Not_Supported]
Sync_Phase_Delay (95)	[Success_Response, Attribute_Not_Supported]
Ramp_rate_Increment (96)	[Success_Response, Attribute_Not_Supported]
Subclass (99)	[Attribute_Not_Settable]

RF Power Generator (20 hex)	**Subclass = 4, Frequency Control**
Attribute Name (ID)	**[Expected Values, Responses]**
Minimum_Frequency (92)	[Attribute_Not_Settable, Attribute_Not_Supported]
Maximum_Frequency (93)	[Attribute_Not_Settable, Attribute_Not_Supported]
Default_Frequency (94)	[Success_Response, Attribute_Not_Supported]
Frequency_Low_Limit (95)	[Success_Response, Attribute_Not_Supported]
Frequency_High_Limit (96)	[Success_Response, Attribute_Not_Supported]

RF Power Generator (20 hex)	**Subclass = 4, Frequency Control**
Attribute Name (ID)	**[Expected Values, Responses]**
Subclass (99)	[Attribute_Not_Settable]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	**[Expected Responses]**
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response, Service_Not_Supported (if no attributes are supported)]
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (code)	**[Expected Responses]**
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	**[Expected Responses]**
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	**[Expected Responses]**
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	**[Expected Responses]**
Object–specific Codes (75..99)	[Service_Not_Supported (x08)] **Note 5**

Note 5: These subclass specific services are defined and limited by the Device Profile

Process Control Valve (1D)	**SubClass = 1, PID and Source Select**

Service Name (Code)	[Expected Responses]
Calibrate (99)	[Success_Response, Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) For each attribute that is implemented with Set access,

- Request a Set_Attribute_Single service, Data_Type = (0x00..0xFF, except 0xC3, 0xCA).
- Request a Set_Attribute_Single service, Data_Units = (0x0000, 0x1000).
- Request a Set_Attribute_Single service, Alarm_Enable = 254.
- Request a Set_Attribute_Single service, Warning_Enable = 254.
- Request a Set_Attribute_Single service, Safe_State = 4..127.
- Request a Set_Attribute_Single service, Process_Variable_Source = 4..255.
  Pass: For each attribute, Error_Response 94 09 FF, Invalid_Attribute_Value.

5.2) Service Test, for implemented services,

- Request a Calibrate service, with parameter = 2..255
  Pass: Error_Response 94 20 FF, Invalid_Parameter.

5.3) Output Limit Maximum, (for RF/DC Generator subclass)

- Request a Set_Attribute_Single, Output_Limit, value = beyond Output_Max.
  Pass: Error_Response 94 09 FF, Invalid_Attribue_Value.

5.4) Frequency Limit Test, (for Frequency Control subclass)

- Request a Set_Attribute_Single, Frequency_Low_Limit, value = beyond Minimum_Frequency.
  Pass: Error_Response 94 09 FF, Invalid_Attribue_Value.
- Request a Set_Attribute_Single, Default_Frequency, value = beyond Minimum_Frequency.
  Pass: Error_Response 94 09 FF, Invalid_Attribue_Value.
- Request a Set_Attribute_Single, Frequency_Low_Limit, value = beyond Maximum_Frequency.
  Pass: Error_Response 94 09 FF, Invalid_Attribue_Value.
- Request a Set_Attribute_Single, Default_Frequency, value = beyond Maximum_Frequency.
  Pass: Error_Response 94 09 FF, Invalid_Attribue_Value.

6) Behavior tests.

Note: Only implemented attributes and services are tested in the Behavior test

6.1) Attribute Access when S-Analog Sensor is in Executing state.

- Request an S-Device Supervisor Start service.
- Request a Get_Attribute_Single, S-Device Supervisor, Device_Status.
  If Device_Status does not = 4, Executing., skip to step 6.2.
- Request a Set_Attribute_Single, Data_Type.
  Pass: Error_Response, 94 0C ff, Object State Conflict.
- Request a Set_Attribute_Single, CV_Data_Type.

Pass: Error_Response, 94 0C ff, Object State Conflict.

- Request a Set_Attribute_Single, Process_Variable.
  Pass: Success_Response
- Request a Set_Attribute_Single, Control_Variable.
  Pass: Success_Response

6.2) Data_Type configuration for Assembly Instances, MFC, V/PG and PCV

MFC - input/output assembly instances, integer = 6/7, floating point = 18/19
V/PG - input assembly instances, integer = 2, floating point = 4 (if implemented)
PCV - input/output assembly instances, integer = 2/18, floating point =7/23

- Open an I/O connection
- Configure the produced and consumed connection path to integer data type assembly instance 6.
- Request a Set_Attribute_Single, I/O EPR = 1000ms.
- Verify the Produced and Consumed Connection Sizes.
  Pass: Produced and Consumed Sizes = Connection Defaults.
- Request a Set_Attribute_Single, Data_Type.
  Pass: Error_Response, 94 0C ff, Object State Conflict.
- Close/Release the I/O connection.
- Open an I/O connection.
- Configure the produced and consumed connection path to floating type assembly instance.
- Verify the Produced and Consumed Connection Sizes.
  Pass: Produced and Consumed Sizes = Floating Point Defaults.
- Request a Set_Attribute_Single, I/O EPR = 1000ms.
- Request a Get_Attribute_Single, Data_Type.
  Pass: Success_Response, 0xCA (REAL).
- Close/Release the I/O connection.
- Reset the device (Identity Reset type 1).
- Open an I/O connection.
- Verify the Produced and Consumed Connection Sizes.
  Pass: Produced and Consumed Sizes = Floating Point Defaults.
- Configure the produced and consumed connection path to integer data type assembly instance 6.
- Close/Release the I/O connection.

6.3) Control_Mode Test.

To be completed in a future revision.

6.4) Process_Variable/Control_Variable Test.

To be completed in a future revision.

6.5) Alarm/Warning Test.

To be completed in a future revision.

6.6) Safe_State/Safe_Value Test.

To be completed in a future revision.

6.7) Ramp_Rate Test.

To be completed in a future revision.

6.8) PID & Source Select Test.

6.8.1) Calibrate Test.

- Request a Calibrate, parameter = 1, Start.
  Pass: Success_Response
- Request a Calibrate, parameter = 1, Start.
  Pass: Error_Response, 94 0B FF, Already in Requested Mode/State
- Request a Get_Attribute_Single, Calibration State.
  Pass: Success_Response, state = 1, Calibrating.
- Request a Calibrate, parameter = 0, Stop
  Pass: Success_Response
- Request a Calibrate, parameter = 0, Stop
  Pass: Error_Response, 94 0B FF, Already in Requested Mode/State
- Request a Get_Attribute_Single, Calibrattion State.
  Pass: Success_Response, state = 0, Controlling.

6.8.2) Check for Crossover Attributes.

- Request a Set_Attribute_Single, Process_Variable_Source = 3.
  Pass: If 3 is valid, Success_Response
- Verify that both the Sensor Crossover High and Low attributes are supported..
  Pass: If 3 is not valid, Error_Response, 94 09 FF, Invalid Attribute Value

6.9) Non-Volatile attributes Test.

- Request a Set_Attribute_Single for each settable attribute, non-default value.
- Request a Get_Attribute_Single for each implemented attribute.
- Request an Identity Object Reset, type = 0.
- When the Network Access is complete, request a Get_Attribute_Single for each implemented non-volatile attribute.
  Pass: Each value = pre-Reset value.

6.10) If Instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.37 S-Gas Calibration Object Test x34

This section defines the conformance test for the S-Gas Calibration object. This test is required when the S-Gas Calibration object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the S-Gas Calibration object.
2) Instance attributes access rules for the S-Gas Calibration object.
3) Common Service implementations for class and instance.
4) Object–specific and Reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
6) Object behavior accessible from the network.
6.1) Gas Calibration Instance implementation.
6.2) non-volatile attributes behavior.
6.3) implementation of the Attributes_List attribute.

Procedure Definition

- Initialization
  The S-Gas Calibration object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection, EPR = 2500 ms.
- Request a Get_Attribute_Single, S-Device Supervisor, Device_Status.
- If Device_Status does not = Idle, request an S-Device Supervisor Reset service.
  Request a Get_Attribute_Single, S-Device Supervisor, Device_Status. Save the Device_Status.

1) Class attributes access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(6..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(4..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..98)	[Service_Not_Supported, Attribute_Not_Supported]
Subclass (99)	[UINT (0), Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	**[Expected Values, Responses]**
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attributes access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	**[Expected Values, Responses]**
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[USINT, Attribute_Not_Supported] **Note 1**
Attributes_List (02)	[USINT[Number_of_Attributes], Attribute_Not_Supported]
Gas_Standard_Number (03)	[UINT(0)]
Valid_Sensor_Instance (04)	[UINT(0)]
Gas_Symbol (05)	[SHORT_STRING, Attribute_Not_Supported
Full_Scale (06)	[(REAL(0), UINT(0)), Attribute_Not_Supported]
Additional_Scalar (07)	[REAL(1.0), Attribute_Not_Supported]
Calibration_Date (08)	[DATE, Attribute_Not_Supported]
Calibration_Gas_Number (09)	[UINT(0), Attribute_Not_Supported]
Gas_Correction_Factor (10)	[REAL(1.0), Attribute_Not_Supported]
Undefined (11..98)	[Attribute_Not_Supported] **Note 2**
Subclass (99)	[UINT(0), Attribute_Not_Supported] **Note 1**

Note 1: Values for these attributes are defined and limited by the Device Profile

Note 2: Values for these attributes are defined and limited by the Device Profile

Mass Flow Controller (1A)	
Attribute Name (ID)	**[Expected Values]**
Number_of_Attributes (01)	(4..12)
Subclass (99)	(1)

Mass Flow Controller (1A)	
Attribute Name (ID)	**[Expected Values]**
Calibration_Pressure (95)	[REAL(101.32), Attribute_Not_Supported]
Calibration_Temperature(96)	[REAL(0), Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	**[Expected Responses]**
Undefined (00)	[Attribute_Not_Supported (x14)]

Attribute Name (ID)	[Expected Responses]
Num_Attributes (01)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes_List (02)	[Attribute_Not_Settable, Attribute_Not_Supported]
Gas_Standard_Number (03)	[Success_Response, Attribute_Not_Settable]
Valid_Sensor_Instance (04)	[Attribute_Not_Settable]
Gas_Symbol (05)	[Success_Response, Attribute_Not_Supported]
Full_Scale (06)	[Attribute_Not_Settable, Attribute_Not_Supported]
Additional_Scalar (07)	[Success_Response, Attribute_Not_Supported]
Calibration_Date (08)	[Attribute_Not_Settable, Attribute_Not_Supported]
Calibration_Gas_Number (09)	[Attribute_Not_Settable, Attribute_Not_Supported]
Gas_Correction_Factor (10)	[Attribute_Not_Settable, Attribute_Not_Supported]
Undefined (11..98)	[Attribute_Not_Supported] **Note 3**
Subclass (99)	[Attribute_Not_Settable, Attribute_Not_Supported]

Note 3: Values for these attributes are defined and limited by the Device Profile

Mass Flow Controller (1A)	
Attribute Name (ID)	**[Expected Values]**
Calibration_Pressure (95)	[Attribute_Not_Settable, Attribute_Not_Supported]
Calibration_Temperature(96)	[Attribute_Not_Settable, Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response, Service_Not_Supported (if no attributes are supported)]
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Get_All_Instances (75)	[Success_Response]
Object–specific Codes (76..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.
- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..104)	[Success_Response , Service_Not_Supported] Profile Specific
Reserved Codes (105..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

No additional Error Tests required.

6) Behavior tests.

6.1) Gas Calibration Instance Test.

- Request a Get_All_Instances service.
  Pass: Success_Response and list of instances.
- Request Get_Attribute_Single Gas_Std_Number, for each Instance in the list.
  Pass: Success_Response.
  Pass: Gas_Std_Number from instance = Gas_Std_Number in list.
- Request Get_Attribute_Single Valid_Sensor_Instance, for each Instance in the list.
  Pass: Success_Response and instance X.
  Pass: instance X = Valid_Sensor_Instance in list.
- If X > 0, Request Get_Attribute_Single S-Analog Sensor, instance X, Value.
  Pass: Success_Response.

6.2) Non-Volatile attributes Test.

- Request a Set_Attribute_Single for each settable attribute, non-default value.
- Request a Get_Attribute_Single for each implemented attribute.
- Request an Identity Object Reset, type = 0.

- When the Network Access is complete, request a Get_Attribute_Single for each implemented non-volatile attribute.
  Pass: Each value = pre-Reset value.

6.3) If Instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.38 Trip Point Object Test x35

This section defines the conformance test for the Trip Point object. This test is required when the Trip Point object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Trip Point object.
2) Instance attributes access rules for the Trip Point object.
3) Common Service implementations for class and instance.
4) Object–specific and Reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
5.1) response when Set_Attribute_Single is requested with invalid data.
6) Object behavior accessible from the network.
6.1) High (or Low) Trip Point behavior.
6.2) Hysteresis Behavior.
6.3) non-volatile attributes behavior.
6.4) implementation of the Attributes_List attribute.

Procedure Definition

- Initialization
  The Trip Point object must be available. Log the object name and test revision.
- Message Connection
  Establish an Explicit Messaging Connection, EPR = 2500 ms.
- Request a Get_Attribute_Single, S-Device Supervisor, Device_Status.
- If Device_Status does not = Executing, request an S-Device Supervisor Start service.
  Request a Get_Attribute_Single, S-Device Supervisor, Device_Status. Save the Device_Status.

1) Class attributes access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1..2), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(6..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(15..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (08..98)	[Service_Not_Supported, Attribute_Not_Supported]
Subclass (99)	[UINT (0), Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attributes access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number_of_Attributes (01)	[USINT, Attribute_Not_Supported] ***Note 1***
Attributes_List (02)	[USINT[Number_of_Attributes], Attribute_Not_Supported]
High_Trip_Point (03)	[INT(0) or defined by Data_Type, Attribute_Not_Supported] ***Note 1***
High_Trip_Enable (04)	[BOOL, Attribute_Not_Supported] ***Note 1***
Low_Trip_Point (05)	[INT(0) or defined by Data_Type , Attribute_Not_Supported] ***Note 1***
Low_Trip_Enable (06)	[BOOL, Attribute_Not_Supported] ***Note 1***
Output_Status (07)	[BOOL]
Polarity (08)	[BOOL, Attribute_Not_Supported]
Override (09)	[USINT(0), Attribute_Not_Supported]
Hysteresis (10)	[INT(0) or defined by Data_Type , Attribute_Not_Supported]
Delay (11)	[UINT(0), Attribute_Not_Supported]
Destination (12)	[EPATH]
Output (13)	[BOOL]
Source (14)	[EPATH]
Input (15)	[INT(0) or defined by Data_Type]
Data_Units (16)	[ENGUNITS, Attribute_Not_Supported]
Data_Type (17)	[USINT(0xCA), Attribute_Not_Supported]
Trip_Point_High_Hysteresis (18)	[INT(0) or defined by Data_Type, Attribute_Not_Supported] ***Note 1***
Trip_Point_Low_Hysteresis (19)	[INT(0) or defined by Data_Type, Attribute_Not_Supported] ***Note 1***
Trip_Status (20)	[USINT, Attribute_Not_Supported] ***Note 1***
Undefined (21..98)	[Attribute_Not_Supported]
Subclass (99)	[UINT(0), Attribute_Not_Supported] ***Note 1***

Note 1: Values for these attributes are defined and limited by the Device Profile

Vacuum/Pressure Gage (1B)	
Attribute Name (ID)	**[Expected Values, Responses]**
Number_of_Attributes (01)	(5..17)

Vacuum/Pressure Gage (1B)	
Attribute Name (ID)	**[Expected Values, Responses]**
High_Trip_Point (03)	Required if Low_Trip_Point is NOT implemented
High_Trip_Enable (04)	Required if High_Trip_Point is implemented
Low_Trip_Point (05)	Required if High_Trip_Point is NOT implemented
Low_Trip_Enable (06)	Required if Low_Trip_Point is implemented
Subclass (99)	(0)

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	**[Expected Responses]**
Undefined (00)	[Attribute_Not_Supported (x14)]
Num_Attributes (01)	[Attribute_Not_Settable (x0E), Attribute_Not_Supported]
Attributes_List (02)	[Attribute_Not_Settable, Attribute_Not_Supported]
High_Trip_Point (03)	[Success_Response, Attribute_Not_Supported] ***Note 1***
High_Trip_Enable (04)	[Success_Response, Attribute_Not_Supported] ***Note 1***
Low_Trip_Point (05)	[Success_Response, Attribute_Not_Supported] ***Note 1***
Low_Trip_Enable (06)	[Success_Response, Attribute_Not_Supported] ***Note 1***
Output_Status (07)	[Attribute_Not_Settable]
Polarity (08)	[Success_Response, Attribute_Not_Supported]
Override (09)	[Success_Response, Attribute_Not_Supported]
Hysteresis (10)	[Success_Response, Attribute_Not_Supported]
Delay (11)	[Success_Response, Attribute_Not_Supported]
Destination (12)	[Success_Response]
Output (13)	[Attribute_Not_Settable]
Source (14)	[Success_Response]
Input (15)	[Success_Response]
Data_Units (16)	[Attribute_Not_Settable, Attribute_Not_Supported]
Data_Type (17)	[Attribute_Not_Settable, Attribute_Not_Supported]
Trip_Point_High_Hysteresis (18)	[INT(0) or defined by Data_Type, Attribute_Not_Supported] ***Note 1***
Trip_Point_Low_Hysteresis (19)	[INT(0) or defined by Data_Type, Attribute_Not_Supported] ***Note 1***
Trip_Status (20)	[USINT, Attribute_Not_Supported] ***Note 1***
Undefined (21..98)	[Attribute_Not_Supported]
Subclass (99)	[Attribute_Not_Settable, Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	**[Expected Responses]**
Reserved (00)	[Service_Not_Supported (x08)]

Service Name (Code)	[Expected Responses]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response, Service_Not_Supported (if no supported attributes)]
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.
- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..104)	[Success_Response , Service_Not_Supported] Profile Specific
Reserved Codes (105..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1) For each attribute that is implemented with Set access,

- Request a Set_Attribute_Single service, High_Trip_Enable = 254.
- Request a Set_Attribute_Single service, Low_Trip_Enable = 254.
- Request a Set_Attribute_Single service, Polarity = 254.
- Request a Set_Attribute_Single service, Override = 4..254.
- Request a Set_Attribute_Single service, Data_Units = 0x00ff..0x07ff.
  Pass: For each attribute, Error_Response 94 09 FF, Invalid_Attribute_Value.

6) Behavior tests.

Note: Only implemented attributes and services are tested in the Behavior test
Pressure/Vacuum Gage profile:
Set the S-Device Supervisor object state = executing.

6.1) High/Low Trip Point behavior.

- Get the Source and Destination attributes
- Verify that the Source and Destination are two bytes long, byte 1 = 0x24.
- Get the Polarity attribute value; use it to determine the output value.
- Get the Delay attribute value; use it for Delay in test procedure.

6.1.1) Low Trip Test.

Note: This test requires the minimum and maximum values of the High and Low Trip Point attributes to be used to set the Trip Point values in the test.

- Set the High Trip Enable = 0.
- Set the Low Trip Point = Low Trip Point maximum.
- Set a test timer = the value of the Delay attribute, wait for timer to expire.
- Get the Status and the Output.
  Pass: Status = Output = TRUE.
- Get the value of the Destination.
  Pass: Output = Destination.
- Set the Low Trip Point = Low Trip Point minimum
- Set a test timer = the value of the Delay attribute, wait for timer to expire.
- Get the Status and the Output.
  Pass: Status = Output = FALSE.
- Get the value of the Destination.
  Pass: Output = Destination.
- Set the High Trip Enable = 1.

6.1.2) High Trip Test.

- Set the Low Trip Enable = 0.
- Set the High Trip Point = High Trip Point minimum.
- Set a test timer = the value of the Delay attribute, wait for timer to expire.
- Get the Status and the Output.
  Pass: Status = Output = TRUE.
- Get the value of the Destination.
  Pass: Output = Destination.
- Set the High Trip Point = High Trip Point maximum.

- Set a test timer = the value of the Delay attribute, wait for timer to expire.
- Get the Status and the Output.
  Pass: Status = Output = FALSE.
- Get the value of the Destination.
  Pass: Output = Destination.
- Set the Low Trip Enable = 1.

6.1.3) Delay Timer Test.

- set the value of the Delay attribute = 100ms.
- Repeat step 6.1.1 and 6.1.2
- set the Delay attribute = 0.

6.1.4) High/Low Trip Disabled Test.

- set the Low Trip Point Enable = 0.
- Repeat step 6.1.1 and 6.1.2, Status and Output will remain FALSE
- set the Low Trip Point Enable = 1.
- set the High Trip Point Enable = 0.
- Repeat step 6.1.1 and 6.1.2, Status and Output will remain FALSE
- set the High Trip Point Enable = 1.

6.2) Hysteresis behavior.

To be developed.

6.3) Non-Volatile attributes Test.

- Request a Set_Attribute_Single for each settable attribute, non-default value.
- Request a Get_Attribute_Single for each implemented attribute.
- Request an Identity Object Reset, type = 0.
- When the Network Access is complete, request a Get_Attribute_Single for each implemented non-volatile attribute.
  Pass: Each value = pre-Reset value.

6.4) If Instance attribute 2, Attribute_List, is implemented,

- Request a Get_Attribute_Single, Number_of_Attributes.
- Request a Get_Attribute_Single, Attributes_List.
  Pass: Success_Response and List size = Number_of_Attributes.
- Request a Get_Attribute_Single for each attribute in the list.
  Pass: Success_Response and value for each attribute.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.39 File Object Test x37

This section defines the File object conformance test. This test is required when the File object is implemented.

Functional Description:

This test verifies the:

1) Class attributes access rules for the File object.
2) Instance attributes access rules for the File object.
3) Common service implementations for the class and instance.
4) Object-specific and Reserved services for the class and instance.
5) Conformance when incorrect data is used to access attributes and services.

5.1) response when Set_Attribute_Single is requested with invalid data.
5.2) response when services are requested with invalid data.
6) Object behavior accessible from the network.
6.0) Dynamically Created Object behavior
6.1) Public Instances Test
6.2) Initialize Tests.
6.3) File Download behavior
6.4) File Upload behavior
6.5) Partial Write Test
6.6) Partial Read Test
6.7) Clear Test

Procedure Definition:

- Initialization
  Log the object name and object and object test software revision and version identifier.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 5000 ms.

1) Class attributes access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT (2]
Max_Instance (02)	[UINT (1..65535), Attribute_Not_Supported]
Num_Instances (03)	[UINT (0..65535)]
Optional_Attribute_List (04)	[(UINT, UINT[size]), size is first UINT, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), size is first UINT, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(6..199), Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(9..199), Attribute_Not_Supported]
Reserved (08..31)	[Attribute_Not_Supported]
Directory (32)	[(UINT, STRINGI, STRINGI)[Num_Instances]]

Undefined (33..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attributes access rules test – run for each instance.

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
State (01)	[USINT(0..7)]
Instance_Name (02)	[STRINGI]
File_Format_Version (03)	[UINT]
File_Name (04)	[STRINGI]
File_Revision (05)	[USINT, USINT]
File_Size (06)	[UDINT]
File_Checksum (07)	[UINT]
Invocation_Method (08)	[USINT(0..4, 100..199, 255)]
File_Save_Parameters (09)	[BYTE(000x000x)] 0 = must be zero, x = 0 or 1
File_Access_Rule (10)	[USINT(0,1)]
File_Encoding_Format (11)	[USINT(0, 1, 100..199, 255)]
Undefined (12..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
State (01)	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
Instance_Name (02)	[Service_Not_Supported, Attribute_Not_Settable]
Instance_Format_Version (03)	[Service_Not_Supported, Attribute_Not_Settable]
File_Name (04)	[Success_Response, Service_Not_Supported, Attribute_Not_Settable]
File_Revision (05)	[Service_Not_Supported, Attribute_Not_Settable]
File_Size (06)	[Service_Not_Supported, Attribute_Not_Settable]
File_Checksum (07)	[Service_Not_Supported, Attribute_Not_Settable]

Attribute Name (ID)	[Expected Values, Responses]
Invocation_Method (08)	[Service_Not_Supported, Attribute_Not_Settable]
File_Save_Parameters (09)	[Service_Not_Supported, Attribute_Not_Settable]
File_Access_Rule (10)	[Service_Not_Supported, Attribute_Not_Settable]
File_Encoding_Format (11)	[Service_Not_Supported, Attribute_Not_Settable]
Undefined (12..99)	[Service_Not_Supported, Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
(00..07)	[Service_Not_Supported (x08)]
Create (08)	[Success_Response, Service_Not_Supported]
(09..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response]
(15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (Code)	[Expected Response]
(00..05)	[Service_Not_Supported (x08)]
Start (06)	[Success_Response, Service_Not_Supported]
Stop (07)	[Success_Response, Service_Not_Supported]
(08)	[Service_Not_Supported]
Delete (09)	[Success_Response, Service_Not_Supported]
(10..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Service_Not_Supported, Attribute_Not_Supported, Attribute_Not_Settable (x0E)]
Reserved (17..20)	[Service_Not_Supported]
Restore (21)	[Success_Response, Service_Not_Supported]
Save (22)	[Success_Response, Service_Not_Supported]
Reserved (23)	[Service_Not_Supported]
Get_Member (24)	[Success_Response, Service_Not_Supported]
Reserved (25..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]

Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Initiate_Upload (75)	[Success_Response]
Initiate_Download (76)	[Success_Response, Service_Not_Supported (x08)]
Initiate_Partial_Read (77)	[Success_Response, Service_Not_Supported]
Initiate_Partial_Write (78)	[Success_Response, Service_Not_Supported]
Upload_Transfer (79)	[Success_Response]
Download_Transfer (80)	[Success_Response, Service_Not_Supported]
Clear (81)	[Success_Response, Service_Not_Supported]
(82..99)	[Service_Not_Supported]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests -- run for each instance.

5.1) If attribute File_Name is settable by Set_Attribute_Single service, test attribute accessible by Set_Attribute_Single using invalid data.

- Attribute 4, File_Name STRINGI with invalid language code.
- Attribute 4, File_Name STRINGI with invalid string data type.
- Attribute 4, File_Name STRINGI with missing string value.
  Pass: Error_Response, 0x09, Invalid Attribute Value, for each test, or
  Pass: Error_Response, 0x20, Invalid Parameter

5.2) Services Invalid Data Test, for implemented services.

- Request an Initiate_Upload with Maximum Transfer Size = 0
  Pass: Errror_Response, 0x20 0x08, Fail on transfer – zero size or 0C FF Object state conflict if file state is File Empty.
- Request an Initiate_Download if the service is supported, File Size = MAX_UDINT (invalid).
  Pass: Error_Response, 0x20 0x04, File Size too large
- Request an Initiate_Download if the service is supported, File Format Version = 0xffff(invalid).
  Pass: Error_Response, 0x20 0x05, File Format Version not Compatible.

- Request an Initiate_Download if the service is supported, File Name is Empty.
  Pass: Error_Response, 0x20 0x0A, File Name empty.
- Request an Initiate_Partial_Read if the service is supported, File_Offset = MAX_UDINT, Read_Size = 100
  Pass: Error_Response, 0x20 0x02, File Offset out of range
- Request an Initiate_Partial_Read if the service is supported, File_Offset = 0, Read_Size = MAX_UDINT
  Pass: Error_Response, 0x20 0x03, Read/Write Size goes beyond end of file
- Request an Initiate_Partial_Write if the service is supported, File_Offset = MAX_UDINT, Read_Size = 100
  Pass: Error_Response, 0x20 0x02, File Offset out of range
- Request an Initiate_Partial_Write if the service is supported, File_Offset = 0, Write_Size = MAX_UDINT
  Pass: Error_Response, 0x20 0x03, Read/Write Size goes beyond end of file
- Request a Get_Member if the service is supported, MemberId = 0.
  Pass: Error_Response, 0x28, Invalid Member Id
- Request a Set_Member if the service is supported, MemberId = 0.
  Pass: Error_Response, 0x28, Invalid Member Id
- Request a Set_Member if the service is supported, MemberId = 127.
  Pass: Error_Response, 0x28, Invalid Member Id

6) Behavior tests run for each File instance.

6.0) Dynamic Tests - Create

This test is skipped if Create service is not supported.

- Request a Get_Attribute_Single, Class, Num_Instances
  Pass: Success_Response, value => start value
- Request a Create, Instance Name = Test.
  Pass: Success_Response, Instance Id in proper range.
- Request a Create, Instance Name = Test.
  Pass: Error_Response 0x20 0x09, Duplicate File Name.
- Run Instance Attributes Test as described in 2).
- Request a Get_Attribute_Single, Class, Num_Instances
  Pass: Success_Response, value = start value + 1
- Request a Get_Attribute_Single, Class, Directory
  Pass: Success_Response, value = includes Instance name “Test”

6.1) Public Instances Test

- For each Public Instance 202..255, request Get Attribute 1
  Pass: Error_Response 0x16, Object Does not Exist.

6.2) Initialize Tests

- Request a Get_Attribute_Single, File_Access_Rule
  Pass: Success_Response

According to the value of this attribute, verify required services for this instance is implemented.

- Request a Get_Attribute_Single, State
  Pass: Success_Response

  If state = File Empty, verify every attribute in this File instance contains correct value.
  - Perform Download Test if the device supports such service (see 6.3)).
  - If Clear service is implemented, Perform Clear Test after file is downloaded successfully.
  - Recover original file after Clear service.

  If state = File Loaded,
  - Perform Initial_Partial_Read Test if the device supports such service.
  - Perform Upload Test (6.4), verfify the Checksum and the file state should be back to File Loaded.

6.3) Download Test if File Type is Read/Write and Object State is File Empty

- Request an Upload_Transfer.
- Request a Download_Transfer.
- Request a Save.
  Pass: Error_Response, 0x0C Object State Conflict for all requests
- Request an Initiate_Download, File Size = 2048, Format Version, Revision = 1.1, Name = Test.
  Pass: Success_Response, Incremental_Burn, Burn_Time, Transfer_Size
- If Error_Response, Size Too Big, decrease File Size until Success_Response.
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 4, Transfer Download Initiated

State = Transfer Download Initiated

- Request an Initiate_Upload.
- Request an Initiate_Partial_Read.
- Request an Upload_Transfer.
- Request a Save.
- Request a Restore.
  Pass: Error_Response, 0x0C, Object State Conflict for all requests

Check DownLoad Errors Test:

- Request a Download_Transfer, Transfer Number = 0, Type = Middle.
  Pass: Error_Response,0x 0C, Object State Conflict
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 1, File Empty
- Request an Initiate_Download, File Size = 2048, Format Version, Revison = 1.1, Name = Test.
- Request a Download_Transfer, Transfer Number = 0, Type = Last.
  Pass: Error_Response, 0x0C, Object State Conflict

- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 1, File Empty
- Request an Initiate_Download, File Size = 2048, Format Version, Revison = 1.1, Name = Test.
- Request a Download_Transfer, Transfer Number = 0, Type = first.
  Pass: Success_Response, with correct Transfer Number
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 6, Transfer Download in Progress
- Request a Download_Transfer, Transfer Number = 1, Type = 3, Abort Tranfer.
  Pass: Success_Response, with correct Transfer Number
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 1, File Empty

- Request an Initiate_Download, File Size = 2048, Format Version, Revision = 1.1, Name = Test.
- Request a Download_Transfer, Transfer Number = 0, Type = first.
  Pass: Success_Response, with correct Transfer Number
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 6, Transfer Download in Progress

State = Transfer Download in Progress

- Request an Initiate_Upload.
- Request an Initiate_Partial_Read.
- Request an Upload_Transfer, Transfer Number = 0.
- Request a Save.
- Request a Restore.
  Pass: Error_Response, 0x0C, Object State Conflict for all requests
- Request a Download_Transfer, Transfer Number = 0, Type = first.
  Pass: Error_Response, 0x0C, Object State Conflict
- Request a Download_Transfer, Transfer Number = 2, Type = middle.
  Pass: Error_Response, 0x20 0x06, Out of Sequence
- Request a Download_Transfer, Transfer Number = 1, Type = first.
  Pass: Error_Response, 0x20 0x06, Out of Sequence
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 1, File Empty
- Re-initialize the download and download first transfer, repeat first packet, all should success.
- Request a Download_Transfer, Transfer Number = 2, Type = first.
  Pass: Error_Response, 0x20 0x06, Out of Sequence
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 1, File Empty

- Re-initialize the download and transfer up to the last type (first and all middle types)
- Request a Download_Transfer, Transfer Number = N, Type = last, Checksum = missing.
  Pass: Error_Response,0x D0, Fail on Checksum
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 1, File Empty

- Re-initialize the download and transfer up to the last type (first and all middle types)
  Transfer Size = Small enough to force rollover
- Request a Download_Transfer to transfer last packet, Type = last, Checksum = invalid.
  Pass: Error_Response, 0xD0, Fail on Checksum
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 1, File Empty

- Re-initialize the download and transfer up to the last type (first and all middle types)
  Transfer Size = Server Transfer Size
- Re-send the previous packet (to verify that server discards it)
- Request a Download_Transfer, Transfer Number = N, Type = Last, Checksum = valid.
  Pass: Success_Response, with correct Transfer Number

During above transfer, if File Size > Incremental Burn Size

- Transfer up to the Incremental Burn Size.
- Set a test timer equal to the Incremental Burn Time. Continue when timer expires.
- While waiting, verify server File State = Storing and doing following tests:
- Request an Initiate_Upload.
- Request an Initiate_Download, File Size = 2048, Format Version, Revision = 1.1, Name = Test.
- Request an Initiate_Partial_Read.
- Request an Initiate_Partial_Write.
- Request an Upload_Transfer, Transfer Number = 0.
- Request a Download_Transfer, Transfer Number = 0.
- Request a Save.
- Request a Restore.
  Pass: Error_Response, 0x0C, Object State Conflict for all requests

- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 2, File Loaded

State = File Loaded

- Request an Upload_Transfer, Transfer Number = 0.
- Request a Download_Transfer, Transfer Number = 0.
  Pass: Error_Response, 0x0C, Object State Conflict for all requests

Check Directory Class Attribute

- Request a Get_Attribute_Single to Class attribute Number_of_Instances
  Pass: Success_Response, value includes new instance if created
- Request a Get_Attribute_Single, Class, Directory
  Pass: Success_Response, value = includes File name "Test"
- Request a Set_Attribute_Single, File_Name = "Test2" if File Name attribute is settable.
  Pass: Success_Response
- Request a Get_Attribute_Single, Class, Directory

Pass: Success_Response, value = includes File name “Test2”

If the device supports Initiate_Partial_Write service, run PartialWriteTest (6.5)).

6.3.1) Download File in Single Transfer

- Request a Clear.
  Pass: Success_Response
- Request an Initiate_Upload, size = small file size < server transfer size.
  Pass: Success_Response, File_Size, Transfer_Size
- Request Download_Transfer, type = Type_Only.
  Pass: Success_Response

6.4) Upload Test

- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 2, File Loaded, otherwise skip the test.

State = File Loaded

- Request an Initiate_Upload, size = 255, 127, 63, 31, 15, 7, 1.
  Pass: Success_Response, File_Size, Transfer_Size
- Request an Initiate_Upload, size = N, small enough to force rollover.
  Pass: Success_Response, File_Size, Transfer_Size
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 3, Transfer Upload Initiated

State = Transfer Upload Initiated
Small Transfer Size, Rollover Check

- Request an Initiate_Download, File Size = 2048, Format Version, Revision = 1.1, Name = Test.
- Request an Initiate_Partial_Write.
- Request a Download_Transfer, Transfer Number = 0, Type = first.
- Request a Save.
- Request a Restore.
  Pass: Error_Response, 0x0C, Object State Conflict for all requests
- Request an Upload_Transfer, Transfer Number = 0.
  Pass: Success_Response, Transfer Number = 0, Type = first, data
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 5, Transfer Upload in Progress

State = Transfer Upload in Progress

- Request an Initiate_Download, File Size = 2048, Format Version, Revision = 1.1, Name = test.
- Request an Initiate_Partial_Write.
- Request a Download_Transfer, Transfer Number = 0, Type = first.
- Request a Save.
- Request a Restore.
  Pass: Error_Response,0x 0C, Object State Conflict for all requests

- Request an Upload_Transfer, Transfer Number = 2.
  Pass: Error_Response, 0x20 0x06, Out of Sequence
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 2, File Loaded

- Re-initialize the upload and transfer up to the middle type
  Pass: Success_Response, Transfer Number = 0..N, Type = first/middle, data
- Re-send the most recent transfer packet Transfer Number = N, Type = middle
  Pass: Success_Response, Transfer Number = N, Type = middle, data
- Complete the upload and transfer up to the last type
  Pass: Success_Response, Transfer Number = N, Type = last, data
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 2, File Loaded

Upload, Maximum Transfer Size

- Request an Initiate_Upload, size = 255.
  Pass: Success_Response, File_Size, Transfer_Size
- Complete the upload
  Pass: Success_Response

6.5) Partial Write Test

- Request an Initiate_Partial_Write, File Offset = 0, Size = 256, Format Version, Revision = 1.1.
  Pass: Success_Response, Incremental_Burn, Burn_Time, Transfer_Size
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 4, Transfer Download Initiated

State = Transfer Download Initiated

- Request Download Transfer and transfer up to the last type (first and all middle types)
  Pass: Success_Response and Transfer number
- Re-send the previous packet (to verify that server discards it)
- Request a Download_Transfer, Transfer Number = 0, Type = last, Checksum = valid.
  Pass: Success_Response
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 2, File Loaded
  Pass: Success_Response, State = 7, Storing, if Storing, wait for State = File Loaded

6.6) Partial Read Test

- Request an Initiate_Partial_Read, offset = 0, transfer size = 256, max size = 255.
  Pass: Success_Response, File_Size, Transfer_Size
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 3, Transfer Upload Initiated
- Complete the upload and transfer up to the last type
  Pass: Success_Response, Transfer Number = N, Type = last, data
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 2, File Loaded

6.6) Clear Test, and Clean-Up if Clear service is supported in this instance.

- Request a Clear
  Pass: Success_Response
- Request a Get_Attribute_Single, State
  Pass: Success_Response, State = 1, File Empty
- Delete a Created Instance
  Pass: Success_Response
- Request a Get_Attribute_Single, Class, Directory
  Pass: Success_Response, Instance entry is deleted

4.40 Time Sync Object Test 0x43

This section defines the conformance test for the Time Sync object. This test is required when the Time Sync object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access rules for the Time Sync object.
2) Instance attributes access rules for the Time Sync object.
3) Common Service implementations for class and instance.
4) Object–specific and Reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
5.1) start list of Error Tests.
6) Object behavior accessible from the network.
6.1) start list of Behavior Tests.

Procedure Definition

- Initialization
  The Time Sync object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 5000ms.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT (1..199), UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT (1..99), UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(6..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(0..199), Service_Not_Supported, Attribute_Not_Supported]
Undefined (02..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.
   Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from

instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request. This test also validates the size of all strucutre type attributes.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Attributes 1..15	[Validate size]
Attirbutes 16, 17	[Validate size (0 to 255), Attribute_Not_Supported (x14)]
DomainNumber (18)	0 to 127
Clock Type (19)	(0..31)
20-28	Validate size
Undefined (29..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Attributes 2, 5, 7-12, 19-28	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
Attributes 1, 3, 4, 6, 13-18	[Success_Response, Service_Not_Supported, Attribute_Not_Settable]
Undefined (03..99)	[Service_Not_Supported, Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] if no attributes are supported
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (code)	[Expected Responses]
(01...02)	[Service_Not_Supported (x08)]
Get_Attributes_List (03)	Success
Set_Attributes_list (04)	Success
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]

Service Name (code)	[Expected Responses]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

There are no error tests for the Time Sync object.

6) Behavior tests.

6.1) Non-Volatile attributes Test

- Set NV attributes to values other than the default.
- Type 0 Reset the device
- Request a Get_Attribute_Single for each implemented NV attribute.
  Pass: Each value = previously set value.
- Restore the original value for each NV attribute.

Note: The behavior test for this object is under development. Contact ODVA for details.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.41 Port Object Test xF4

This section defines the Port object conformance test. This test is required for all devices that implement the Port object.

Functional Description

This test verifies the:

1) Class attributes access rules for the Port object.
2) Instance attributes access rules for the Port object.
3) Common Service implementations for class and instance.
4) Object–specific and Reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
6) Object behavior accessible from the network.

Procedure Definition

- Initialization
  The Port object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT (2)]
Max_Instance (02)	[UINT]
Num_Instances (03)	[UINT]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT (1..199), Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT (0..199), Attribute_Not_Supported]
Entry_Port (08)	[UINT]
All_Ports (09)	[(UINT, UINT)[Max_Instance]]
Undefined (10..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.
   Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Port_Type (01)	[(UINT (0..5, 100..199, 65535)]
Port_Number (02)	[UINT]
Port_Object (03)	[UINT(size), PADDED_EPATH] size = number of words
Port_Name (04)	[SHORT_STRING] 64 characters max
Port_Type_Name (05)	[SHORT_STRING, Attribute_Not_Supported] 64 characters max
Port_Description (06)	[SHORT_STRING, Attribute_Not_Supported] 64 characters max
Port Number and Node_Address (07)	[PADDED_EPATH] = value of Port_Number
Port_Node_Range (08)	[(UINT, UINT), Attribute_Not_Supported] **Note 1**
Port_key (09)	[PACKED_EPATH, Attribute_Not_Supported]
Port Routing Capabilities (10)	[DWORD]
Undefined (11..99)	[Attribute_Not_Supported]

Note 1: EtherNet/IP devices shall return Attribute_Not_Supported. Otherwise, this is required.

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Port_Type (01)	[Success_Response, Service_Not_Supported]
Port_Number (02)	[Service_Not_Supported , Attribute_Not_Settable (x0E)]
Port_Object (03)	[Attribute_Not_Settable, Service_Not_Supported]
Port_Name (04)	[Attribute_Not_Settable, Service_Not_Supported]
Port_Type_Name (05)	[Attribute_Not_Settable, Service_Not_Supported]
Port_Description (06)	[Success_Response, Service_Not_Supported, Attribute_Not_Settable, Attribute_Not_Supported]
Node_Address (07)	[Attribute_Not_Settable, Service_Not_Supported]
Port_Node_Range (08)	[Attribute_Not_Settable, Service_Not_Supported]
Port_key (09)	[Attribute_Not_Settable, Service_Not_Supported]
Port Routing Capabilities (10)	[Attribute_Not_Settable, Service_Not_Supported]
Any Attribute (11..99)	[Service_Not_Supported, Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..13)	[Service_Not_Supported]

Service Name (Code)	[Expected Responses]
Get_Attribute_Single (14)	[Success_Response]
Reserved (15..49)	[Service_Not_Supported]

Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
(02..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response]
(15)	[Service_Not_Supported]
Set_Attribute_Single(16)	[Success_Response, Service_Not_Supported]
(17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each of the Reserved Services, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests
None.

6) Behavior Tests.

- For each Port instance, check the Port Type and characteristic is consistent.
- Check the Port Routing Capability of each Port instance. In the future may do behavior test according to its capability.

Delete the connections at the conclusion of this test and leave the device in the On–Line state.

4.42 Base Energy Object Test x4E

This section defines the Base Energy Object conformance test. This test is required for all devices that implement the Base Energy Object.

Functional Description

This test verifies the:

1) Class attributes access rules and implementation rules for the Base Energy Object.
2) Instance attributes access rules and implementation rules for the Base Energy Object.
3) Common Service implementations for class and instance.
4) Object–specific and reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
5.1) Search all implemented settable attributes, set out of range values to these attributes, and verify getting expected status code.
6) Object behavior accessible from the network.
6.1) Check if any conditional class/instance attribute is required for the DUT and verify the DUT supports that correctly.
6.2) Verify any conditional service is required and implemented.
6.3) Behavior test for each Base Energy Object instance
6.4) Class level Reset service test

Procedure Definition

- Initialization
  The Base Energy Object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access all the object supported attributes. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request, or the status code is 0x08 (Service not supported) if none of the class attributes is implemented.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT (2)]
Max_Instance (02)	[UINT, Attribute_Not_Supported]
Num_Instances (03)	[UINT, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT (1..199), Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT (0..199), Attribute_Not_Supported]
Undefined (08-99)	[Attribute_Not_Supported (x14)]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access all object supported attributes. Use data returned by the appropriate Get_Attribute_Single

service.
Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attributes access rules and implementation rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access all object supported attributes. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Resource_Type (01)	[(UINT (0,1,2, 100..199)]
Capability (02)	[WORD (Bit 0-2)]
Energy_Accuracy (03)	[UINT]
Energy_Accuracy_Basis (04)	[UINT (0, 1, 2), Attribute_Not_Supported]
Full_Scale_Reading (05)	[REAL, Attribute_Not_Supported]
Data_Status (06)	[UINT, Attribute_Not_Supported]
Consumed_Energy_Odometer (07)	[Array[5] of UINT, Attribute_Not_Supported]
Generated_Energy_Odometer (08)	[Array[5] of UINT, Attribute_Not_Supported]
Total_Energy_Odometer (09)	[Array[5] of UINT, Attribute_Not_Supported]
Energy_Transfer_Rate (10)	[REAL, Attribute_Not_Supported]
Energy_Transfer_Rate_User_Setting (11)	[REAL, Attribute_Not_Supported]
Energy_Type_Specific_Object_Path (12)	[UINT, Padded EPATH]
Energy_Aggregation_Path_Array_Size(13)	[UINT, Attribute_Not_Supported]
Energy_Aggregation_Paths (14)	[Array of [UINT, Padded EPATH], Attribute_Not_Supported]
Energy_Identifier (15)	[STRINGI, Attribute_Not_Supported]
Odometer_Reset_Enable (16)	[BOOL, Attribute_Not_Supported]
Metering_State (17)	[BOOL, Attribute_Not_Supported]
Extended Data Status (18)	[UINT, Attribute_Not_Supported]
Undefined (19-99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access all object supported attributes. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request, or Service_Not_Supported (x14) if none of the optional Settable attributes are implemented.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Resource_Type (01)	[Attribute_Not_Settable (x0E)]
Capability (02)	[Attribute_Not_Settable (x0E)]

Energy_Accuracy (03)	[Attribute_Not_Settable (x0E)]
Energy_Accuracy_Basis (04)	[Attribute_Not_Supported (x14), Success]
Full_Scale_Reading (05)	[Attribute_Not_Supported (x14), Success]
Data_Status (06)	[Attribute_Not_Settable, Attribute_Not_Supported]
Consumed_Energy_Odometer (07)	[Attribute_Not_Settable, Attribute_Not_Supported]
Generated_Energy_Odometer (08)	[Attribute_Not_Settable, Attribute_Not_Supported]
Total_Energy_Odometer (09)	[Attribute_Not_Settable, Attribute_Not_Supported]
Energy_Transfer_Rate (10)	[Attribute_Not_Settable, Attribute_Not_Supported]
Energy_Transfer_Rate_User_Setting (11)	[Attribute_Not_Supported (x14), Success]
Energy_Type_Specific_Object_Path (12)	[Attribute_Not_Settable]
Energy_Aggregation_Path_Array_Size(13)	[Attribute_Not_Settable, Attribute_Not_Supported]
Energy_Aggregation_Paths (14)	[Attribute_Not_Supported (x14), Success]
Energy_Identifier (15)	Attribute_Not_Supported (x14), Success]
Odometer_Reset_Enable (16)	[Attribute_Not_Supported (x14), Success]
Metering_State (17)	[Attribute_Not_Settable, Attribute_Not_Supported]
Extended Data Status (18)	[UINT, Attribute_Not_Supported]
Undefined (18-99)	[Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.
  If Get_Attributes_All service is implemented, the response is also being validated. The test compares the attribute value returned from this service to the value received from Get_Attribute_Single service. All attribute values should match except for some dynamically changing odometer values.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
Get_Attribute_List (03)	[Success_Response, Service_Not_Supported]
Reset (05)	[Success_Response, Service_Not_Supported]
Create (08)	[Success_Response, Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response, Service_Not_Supported]
Reserved (all others between 0-49)	[Service_Not_Supported]

- ➢ Request each of the Common Services, 0 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.
  If Get_Attributes_All service is implemented, the response is also validated. The test compares the attribute value returned from this service to the value received from Get_Attribute_Single service. For most attributes the values should match. For some changing attributes, such as odometer readings, the values could be different.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
Get_Attribute_List (03)	[Success_Response, Service_Not_Supported]
Set_Attribute_List (04)	[Success_Response, Service_Not_Supported]
Reset (05)	[Success_Response, Service_Not_Supported]
Create (08)	[Success_Response, Service_Not_Supported]
Delete (09)	[Success_Response, Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response]
Set_Attribute_Single (16)	[Success_Response, Service_Not_Supported]
Get_Member (24)	[Success_Response, Service_Not_Supported]
Set_Member (25)	[Success_Response, Service_Not_Supported]
Insert_Member (26)	[Success_Response, Service_Not_Supported]
Remove_Member (27)	[Success_Response, Service_Not_Supported]
Reserved (all others between 0-49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Start_Metering (75)	[Success_Response, Service_Not_Supported]
Stop_Metering (76)	[Success_Response, Service_Not_Supported]
Object–specific Codes (77..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Start_Metering (75)	[Success_Response, Service_Not_Supported]
Stop_Metering (76)	[Success_Response, Service_Not_Supported]
Object–specific Codes (77..99)	[Service_Not_Supported (x08)]

- Request each of the Reserved Services, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests

Search all implemented settable attributes, set out of range values to these attributes, and verify getting expected status code 0x09 (Invalid_Attribute_Value).

6) Behavior Tests

6.1) Conditional attributes check.

Check if any conditional attribute (class level or instance level) is required and implemented.

6.2) Conditional services check.

a. Check if any class attribute is implemented. If yes, verify Get_Attribute_Single class service is also implemented, otherwise return error.
b. Check if Set_Attribute_Single instance service is required and implemented.

6.3) Behavior test for each Base Energy Object instance

6.3.1) Verify Instance Attribute 12 (Energy Type Specific Object Path) points to a valid energy type specific object instance.

6.3.2) Metering results consistency test

Verify the values reported in the Base Energy Object and the energy type specific objects are consistent.

6.3.3) Instance level Reset service test

Verify the behavior of Reset type 0, 1, 2 and invalid values if device supports Reset service.

a. Type 0 Reset expected behavior: 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.
b. Type 1 Reset expected behavior: Reset Base Energy Object and Energy Type-Specific Object pointed to by the Energy Type-specific Object Path to factory defaults.
c. Type 2 Reset expected behavior: Reset the configuration attributes of the Base Energy Object and Energy Type-Specific Object pointed to by the Energy Type Specific Object Path without resetting Energy Odometers.

6.3.4) Object specific services test.

Verify the behavior of Sart_Metering and Stop_Metering services.

6.3.5) Instance level NV attributes behavior test.

Find all implemented NV attributes. Set their values other than the default. Cycle the power of the device and verify the attributes' values retaining during the power cycle.

6.4) Class level Reset service test

6.4.1) The test verifies the device behavior of Reset type 0, 1, 2, and invalid values if Class level Reset service is implemented,

6.4.2) If Class level Reset service is not supported, the test verifies the device returns correct error code.

4.43 <u>Electrical Energy Object Test x4F</u>

This section defines the Electrical Energy Object conformance test. This test is required for all devices that implement the Electrical Energy Object.

Functional Description

This test verifies the:

1) Class attributes access rules and implementation rules for the Electrical Energy Object.
2) Instance attributes access rules and implementation rules for the Electrical Energy Object.
3) Common Service implementations for class and instance.
4) Object–specific and reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.

5.1) Search all implemented settable attributes, set out of range values to these attributes, and verify getting expected status code.

6) Object behavior accessible from the network.

Procedure Definition

- Initialization
  The Electrical Energy Object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access all the object supported attributes. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request, or the status code is 0x08 (Service not supported) if none of the class attributes is implemented.

Attribute Name (ID)	**[Expected Values, Responses]**
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT (2)]
Max_Instance (02)	[UINT, Attribute_Not_Supported]
Num_Instances (03)	[UINT, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT (1..199), Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT (0..199), Attribute_Not_Supported]
Undefined (08-99)	[Attribute_Not_Supported (x14)]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access all object supported attributes. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	**[Expected Responses]**
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attributes access rules and implementation rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access all object supported attributes. Save the returned value for each Get_Attribute_Single request. Electrical Energy Object has 41 attributes.
- **Pass:** The expected response from table below for each Get_Attribute_Single request. The test also verifies the attribute values fall in required data range (See CIP specification Vol 1_3.11 Table 5-52.2 for reference).

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Attribute (01-40, except 3, 27) (Optional)	[Success_Response, Attribute_Not_Supported]
Real_Energy_Net_Odometer (03) (Conditional)	[Array[5] of UINT, Attribute_Not_Supported]
Total_Real_Power (27) (Conditional)	[Real, Attribute_Not_Supported]
Associated_Base_Energy_Object_Path(41)(Required)	[UINT, Padded EPATH]
Undefined (42-99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access all object supported attributes. Use data returned by the appropriate Get_Attribute_Single service. Electrical Energy Object has none settable attribute, so Set_Attribute_Single service is not needed.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported] (x08)]
Attribute (01-41)	[Service_Not_Supported] (x08)]
Undefined (42-99)	[Service_Not_Supported] (x08)]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.
  If Get_Attributes_All service is implemented, the response is also validated. The test compares the attribute value returned from this service to the value received from Get_Attribute_Single service. All attribute values should match.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
Get_Attribute_List (03)	[Success_Response, Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response, Service_Not_Supported]
Reserved (all others between 0-49)	[Service_Not_Supported]

1) Request each of the Common Services, 0 - 49, addressed to a valid Instance ID.
   Pass: The expected response from table below for each Common Service request.

If Get_Attributes_All service is implemented, the response is also being validated. The test compares the attribute value returned from this service to the value received from Get_Attribute_Single service. For most attributes the values should match. For some changing attributes, such as odometer readings, the values could be different.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
Get_Attribute_List (03)	[Success_Response, Service_Not_Supported]
Set_Attribute_List (04)	[Success_Response, Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response]
Set_Attribute_Single (16)	[Service_Not_Supported]
Reserved (all others between 0-49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each of the Reserved Services, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests
Verify there's no settable attributes.

6) Behavior Tests
For each instance of Electrical Energy Object, check following behaviors:

6.1) Instance Attribute 3 (Real Energy Net Odometer) and attribute 27 (Total Real Power) are conditional. At least one of them should be implemented.

6.2) Instance Attribute 41 (Associated Base Energy Object Path): shall be valid and point to the Base Energy Object.

6.3) The value in Attribute 3 (Real Energy Net Odometer) should match the value in the Total Energy Odometer attribute of the generic Base Energy Object that is pointed to by the Associated Base Energy Object Path attribute.

4.44 Non-Electrical Energy Object Test x50

This section defines the Non-Electrical Energy Object conformance test. This test is required for all devices that implement the Non-Electrical Energy Object.

Functional Description

This test verifies the:

1) Class attributes access rules and implementation rules for the Non-Electrical Energy Object.
2) Instance attributes access rules and implementation rules for the Non-Electrical Energy Object.
3) Common Service implementations for class and instance.
4) Object–specific and reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
5.1) Search all implemented settable attributes, set out of range values to these attributes, and verify getting expected status code.
6) Object behavior accessible from the network.

Procedure Definition

- Initialization
  The Non-Electrical Energy Object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access all the object supported attributes. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request, or the status code is 0x08 (Service not supported) if none of the class attributes is implemented.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT (2)]
Max_Instance (02)	[UINT, Attribute_Not_Supported]
Num_Instances (03)	[UINT, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT (1..199), Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT (0..199), Attribute_Not_Supported]
Undefined (08-99)	[Attribute_Not_Supported (x14)]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access all object supported attributes. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attributes access rules and implementation rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access all object supported attributes. Save the returned value for each Get_Attribute_Single request.
- **Pass:** The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Energy_Consumed_Odometer (01)	[Array[5] of UINT, Attribute_Not_Supported]
Energy_Generated_Odometer (02)	[Array[5] of UINT, Attribute_Not_Supported]
Energy_Net_Odometer (03)	[Array[5] of UINT, Attribute_Not_Supported]
Resource_Type (04)	[UINT(0x0800-0x1015)]
Engineering_Unit (05)	[ENGUNIT]
Engineering_Unit_Description (06)	[STRINGI, Attribute_Not_Supported]
Normalization_Constant_Multiplier (07)	[UNIT]
Normalization_Constant_Divisor (08)	[INT]
Energy_Transfer_Rate (09)	[REAL, Attribute_Not_Supported]
Energy_Flow_Rate_Time_Base (10)	[ENGUNIT, Attribute_Not_Supported]
Associated_Base_Energy_Object_Path (11)	[[UINT, Padded EPATH]
Undefined (12-99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access all object supported attributes. Use data returned by the appropriate Get_Attribute_Single service. Non-Electrical Energy Object has none settable attribute, so Set_Attribute_Single service is not needed.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported] (x08)]
Attribute (01-11)	[Service_Not_Supported] (x08)]
Undefined (12-99)	[Service_Not_Supported] (x08)]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.
  If Get_Attributes_All service is implemented, the response is also being validated. The test compares the attribute value returned from this service to the value received from Get_Attribute_Single service. All attribute values should match.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]

Service Name (Code)	[Expected Responses]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
Get_Attribute_List (03)	[Success_Response, Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response, Service_Not_Supported]
Reserved (all others between 0-49)	[Service_Not_Supported]

m) Request each of the Common Services, 0 - 49, addressed to a valid Instance ID.
Pass: The expected response from table below for each Common Service request.
If Get_Attributes_All service is implemented, the response is also validated. The test compares the attribute value returned from this service to the value received from Get_Attribute_Single service. For most attributes the values should match. For some changing attributes, such as odometer readings, the values could be different.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success_Response, Service_Not_Supported]
Get_Attribute_List (03)	[Success_Response, Service_Not_Supported]
Set_Attribute_List (04)	[Success_Response, Service_Not_Supported]
Get_Attribute_Single (14)	[Success_Response]
Set_Attribute_Single (16)	[Service_Not_Supported]
Reserved (all others between 0-49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each of the Reserved Services, 100 - 255, addressed to a valid Instance ID.
Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests

Verify there's no settable attributes.

6) Behavior Tests

For each instance of Non-Electrical Energy Object, check following behaviors:

6.1) Instance Attribute 3 (Energy Net Odometer) and attribute 9 (Energy Transfer Rate) are conditional. At least one of them should be implemented.

6.2) Instance Attribute 11 (Associated Base Energy Object Path): shall be valid and point to a Base Energy Object.

6.3) The value in Attribute 3 Energy Net Odometer, if implemented, should match the value in the Total Energy Odometer attribute of the generic Base Energy Object that is pointed to by the Associated Base Energy Object Path attribute.

4.45 Motion Device Axis Object Test x42

This section defines the Motion Device Axis Object (MDAO) conformance test. This test is required for all devices that implement the Motion Device Axis Object. This object is required for CIP Motion devices.

Functional Description

This test verifies the:

1) Class attributes access rules and implementation rules for the Motion Device Axis Object.
2) Instance attributes access rules and implementation rules for the Motion Device Axis Object.
3) Common Service implementations for class and instance.
4) Object–specific and reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
6) Object behavior accessible from the network.

Procedure Definition

- Initialization
  The Motion Device Axis Object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access all the object supported attributes. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT (3)]
Max_Instance (02)	[UINT(1..65535), Attribute_Not_Supported]
Num_Instances (03)	[UINT(1..65535), Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT (1..199), Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT, Attribute_Not_Supported]
Attribute (14–16, 18-20, 34)	[Success]
Attribute (17, 21-22, 28-33, 35)	[Success or Attribute_Not_Supported]
Undefined (08-13, 23-27, 36-99)	[Attribute_Not_Supported (x14)]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access all object supported attributes. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT, Attribute_Not_Settable(0x0E)]

Attribute Name (ID)	[Expected Values, Responses]
Max_Instance (02)	[UINT, Attribute_Not_Settable(0x0E)]
Num_Instances (03)	[UINT, Attribute_Not_Settable(0x0E)]
Optional_Attribute_List (04)	[UINT, Attribute_Not_Settable(0x0E)]
Optional_Service_List (05)	[UINT, Attribute_Not_Settable(0x0E)]
Max_Class_Attribute_Id (06)	[UINT, Attribute_Not_Settable(0x0E)]
Max_Instance_Attribute_Id (07)	[UINT, Attribute_Not_Settable(0x0E)]
Attribute (14–20, 28, 30-31, 34)	[Attribute_Not_Settable(0x0E)]
Attribute (21-22, 29, 32-33, 35)	[Success]
Undefined (08-13, 23-27, 36-99)	[Attribute_Not_Supported (x14)]

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance. If multiple Motion Device Axis Object instances exist in a device, the Conformance Test Tool will run Get Attribute Test, Set Attribute Test, Common Service Test, Object Specific Service Test, Error Test and Behavior Test for each found instance.

2) Instance attributes access rules and implementation rules test.

- Request Get_Attribute_Single service (x0E) addressed to a valid instance to access all object supported attributes. Save the returned value for each Get_Attribute_Single request.

Because of the large number of instance attributes in the Motion Device Axis Object, this document does not enumeratethem here . Instead,the general rules used to validate conditional attribute implementation, valid value enforcement, and Set Access are provided here. Please refer to the specification for attribute details.

Pass criteria:

- ➢ All required attributes shall return Success to Get_Attribute_Single request
- ➢ All implemented Conditional or Optional attributes shall return Success to Get_Attribute_Single request
- ➢ The attribute value in the successful response shall be in the data range specified by the specification.
- ➢ Not implemented attribute shall return Attribute_Not_Supported(0x14)

Request Set_Attribute_Single service (x10) addressed to a valid instance to access all supported attributes. Use the data that was previously returned by the corresponding Get_Attribute_Single service.

Note: According to the MDAO attribute table footnotes as below, those attributes marked as Set * are expected to return Attribute_Not_Supported (0x14) to the Set_Attribute_Single service.
Set * - Indicates the attribute is normally set by the CIP Motion connection data block and not by a Set service.

Pass criteria:

- ➢ All implemented CIP Settable attributes shall return Success to the Set_Attribute_Single service, otherwise return Attribute_Not_Supported (0x14).
- ➢ Not implemented attributes shall return Attribute_Not_Supported (0x14)

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.
  If Get_Attributes_All service is implemented, the response is also validatedby comparing the attribute value returned from this service to the values received from Get_Attribute_Single service. All attribute values should match.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_List (03)	[Success_Response]
Set_Attributes_List (04)	[Success_Response]
Get_Attribute_Single (14)	[Success_Response]
Set_Attribute_Single (16)	[Success_Response]
Group_Sync (28)	[Success_Response, Service_Not_Supported]
Reserved (all others between 0-49)	[Service_Not_Supported]

- Request each of the Common Services, 0 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_List (03)	[Success_Response]
Set_Attributes_List (04)	[Success_Response]
Get_Attribute_Single (14)	[Success_Response]
Set_Attribute_Single (16)	[Success_Response]
Group_Sync (28)	[Service_Not_Supported]
Reserved (all others between 0-49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]

Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 – 99 (0x4B-0x63), addressed to a valid Instance ID. **Pass:** The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Get_Axis_Attributes_List (0x4B)	[Success_Response]
Set_Axis_Attributes_List (04C)	[Success_Response]
Set_Cyclic_Write_List (0x4D)	[Success_Response]
Set_Cyclic_Read_List (0x4E)	[Success_Response]
Run_Motor_Test (0x4F)	[Success_Response, Service_Not_Supported]
Get_Motor_Test_Data (0x50)	[Success_Response, Service_Not_Supported]
Run_Inertia_Test (0x51)	[Success_Response, Service_Not_Supported]
Get_Inertia_Test_Data (0x52)	[Success_Response, Service_Not_Supported]
Run_Hookup_Test (0x53)	[Success_Response]
Get_Hookup_Test_Data (0x54)	[Success_Response]
Reserved (all others between 0x55-0x63)	[Service_Not_Supported]

- Request each of the Reserved Services, 100 - 255, addressed to a valid Instance ID. **Pass:** The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests

There areno error tests at this time.

6) Behavior Tests

There are no behavior tests at this time.

Note: For CIP Motion compliant devices Module Status LED is required. Axis Status LED is also required unless the device is equipped with a multi-character alphanumeric display.

4.46 Originator Connection List Object Test x45

This section defines the Originator Connection List Object conformance test. This test is required for all devices that implement the Originator Connection List Object. This object can be implemented by any device that originator connections. Devices that do not originate connections shall not support this object.

Functional Description

This test verifies the:

1) Class attributes access rules and implementation rules for the Originator Connection List Object.
2) Common Service implementations for class and instance.
3) Object–specific and reserved service implementations for class and instance.
4) Conformance when incorrect data is used to access attributes and services.
5) Object behavior accessible from the network.

Procedure Definition

- Initialization
  The Originator Connection List Object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access all the object supported attributes. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14), Service_Not_Supported(0x08)]
Revision (01)	[UINT (1), Attribute_Not_Supported, Service_Not_Supported(0x08)]
Max_Instance (02)	[UINT(1..65535), Attribute_Not_Supported, Service_Not_Supported(0x08)]
Num_Instances (03)	[UINT(1..65535), Attribute_Not_Supported, Service_Not_Supported(0x08)]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Attribute_Not_Supported, Service_Not_Supported(0x08)]
Optional_Service_List (05)	[(UINT, UINT[size]), Attribute_Not_Supported, Service_Not_Supported(0x08)]
Max_Class_Attribute_Id (06)	[UINT (1..199), Attribute_Not_Supported, Service_Not_Supported(0x08)]
Max_Instance_Attribute_Id (07)	[UINT, Attribute_Not_Supported, Service_Not_Supported(0x08)]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access all object supported attributes. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[[Service_Not_Supported(0x08)]
Revision (01)	[Service_Not_Supported(0x08)]
Max_Instance (02)	[Service_Not_Supported(0x08)]
Num_Instances (03)	[Service_Not_Supported(0x08)]
Optional_Attribute_List (04)	[Service_Not_Supported(0x08)]
Optional_Service_List (05)	[Service_Not_Supported(0x08)]
Max_Class_Attribute_Id (06)	[Service_Not_Supported(0x08)]
Max_Instance_Attribute_Id (07)	[Service_Not_Supported(0x08)]

Note: There's no instance attributes defined for the Originator Connection List Object. So the Get_Attribute_Single Service (0x0E) and Set_Attribute_Single_Service (0x10) should not be implemented.

2) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.
  If Get_Attributes_All service is implemented, the response is also validatedby comparing the attribute value returned from this service to the values received from Get_Attribute_Single service. Attribute 1, 2, 3, 8, 9 values should match.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success, Service_not_Supported]
Create(08)	[Success]
Get_Attribute_Single (14)	[Success, Service_not_Supported]
Reserved (all others between 0-49)	[Service_Not_Supported]

- Request each of the Common Services, 0 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success, Service_not_Supported]
Delete(09)	[Success]
Get_Attribute_Single (14)	[Success, Service_not_Supported]
Reserved (all others between 0-49)	[Service_Not_Supported]

3) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75, 77 - 99, addressed to the class, Instance ID zero.

Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75, 77..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 – 99 (0x4B-0x63), addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Connection Read(76)	[Success]
Object–specific Codes (75,77…99)	[Service_Not_Supported (x08)]

- Request each of the Reserved Services, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

4) Error Tests

4.1) Send a Create request 0x08 at class level with Path from originator properties using reserved bits.

Pass: the DUT shall return 0x20 (Invalid Parameter).

4.2) Send a Create request 0x08 at instance level.

Pass: The DUT returns 0x08 (Service not Supported).

6) Behavior Tests

6.1) Issue a Create request 0x08 with correct parameter
Pass: The DUT creates an OCL instance successfully.

6.2) Test the Connection Read service on the created instance
Pass: The DUT returns valid connection configuration data

6.3) Issue a Delete request 0x08 to the created instance
Pass: The DUT returns Success

4.47 Target Connection List Object Test x4D

This section defines the Target Connection List Object conformance test. This test is required for all devices that implement the Target Connection List Object. This object can be implemented by any device that accepts the ForwardOPen services of the Connection Manager Object.

Functional Description

This test verifies the:

1) Class attributes access rules and implementation rules for the Target Connection List Object.
2) Instance attributes access rules and implementation rules for the Target Connection List Object.
3) Common Service implementations for class and instance.
4) Object–specific and reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
6) Object behavior accessible from the network.

Procedure Definition

- Initialization
  The Target Connection List Object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access all the object supported attributes. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Attribute_Not_Supported]
Max_Instance (02)	[UINT(1..65535)]
Num_Instances (03)	[UINT(1..65535)]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT (1..199), Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT, Attribute_Not_Supported]
Connection Data Set Revision	[UINT(0..65535), Success]
Capabilities	[DWORD(1..65535), Success]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access all object supported attributes. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]

Attribute Name (ID)	[Expected Values, Responses]
Revision (01)	[UINT, Service_Not_Supported(0x08)]
Max_Instance (02)	[UINT, Service_Not_Supported(0x08)]
Num_Instances (03)	[UINT, Service_Not_Supported(0x08)]
Optional_Attribute_List (04)	[UINT, Service_Not_Supported(0x08)]
Optional_Service_List (05)	[UINT, Service_Not_Supported(0x08)]
Max_Class_Attribute_Id (06)	[UINT, Service_Not_Supported(0x08)]
Max_Instance_Attribute_Id (07)	[UINT, Service_Not_Supported(0x08)]
Connection Data Set Revision	[UINT, Service_Not_Supported(0x08)]
Capabilities	[DWORD, Service_Not_Supported(0x08)]

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance. If multiple Target Connection List Object instances exist in a device, the Conformance Test Tool will run Get Attribute Test, Set Attribute Test, and Common Service Test for each found instance.

The found instances shall be in range 1 – 5, or 100 – 65535.

2) Instance attributes access rules and implementation rules test.

- Request Get_Attribute_Single service (x0E) addressed to a valid instance to access all object supported attributes. Save the returned value for each Get_Attribute_Single request.

  If any instance attributes are supported, then all of the instance attributes shall be supported.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
O->T RPI Range (01)	[Struct of UDINT(0..0xffffffff),UDINT(0..0xffffffff), Attribute_Not_Supported]
T->O RPI Range (02)	[Struct of UDINT(0..0xffffffff),UDINT(0..0xffffffff), Attribute_Not_Supported]
Trigger, Transport and Application Type (03)	[DWORD, Attribute_Not_Supported]
Connection Parameters (04)	[DWORD, Attribute_Not_Supported]
O->T Size (05)	[Struct of UINT(0..65535), UINT(0..65535)), Attribute_Not_Supported]
T->O Size (06)	[Struct of UINT(0..65535), UINT(0..65535)), Attribute_Not_Supported]
PIT Range (07)	[Struct of UDINT(0..0xffffffff),UDINT(0..0xffffffff), Attribute_Not_Supported]
Connection Name String	[SHORT_STRING, Attribute_Not_Supported]

Request Set_Attribute_Single service (x10) addressed to a valid instance to access all supported attributes. Use the data that was previously returned by the corresponding Get_Attribute_Single service.
Each attribute shall return Attribute_Not_Supported (0x14) or Service_Not_Supported (0x08).

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.
  If Get_Attributes_All service is implemented, the response is also validatedby comparing the

attribute value returned from this service to the values received from Get_Attribute_Single service. Attribute 1, 2, 3, 8, 9 values should match.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success, Service_not_Supported]
Get_Attribute_Single (14)	[Success_Response]
Reserved (all others between 0-49)	[Service_Not_Supported]

- Request each of the Common Services, 0 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success, Service_not_Supported]
Get_Attribute_Single (14)	[Success, Service_not_Supported]
Reserved (all others between 0-49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75, 77 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75, 77..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 – 99 (0x4B-0x63), addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each of the Reserved Services, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests

5.1) Send a Connection Read request 0x4C at class level Connection Specifier is in range 6 – 99.
Pass: the DUT shall return 0x20 (Invalid Parameter).

5.2) Send a Connection Read request 0x4C at instance level.
Pass: The DUT returns 0x08 (Service not Supported).

6) Behavior Tests

Test class level Connection Read service.

6.1) Issue a Connection Read request 0x4C with Connection Specifier = 1 and Connection Structure Exclusion = 1 (Connection Name String should be omitted).
6.2) Issue a Connection Read request 0x4C with Connection Specifier = 1 and Connection Structure Exclusion = 0 (Connection Name String should not be omitted). Compare response data are the same as above except Connection Name String.
If the device return status 0x011 (Reply data too large), invoke a Large_Forward_Open to make a new Explicit Connection with the Additional Status Word in the status 0x11 response as the T->O size. Resend 0x4C request.

Pass: The DUT responds success 0x00 with Additional Status word 0x0000 or 0x0001 to 0x4C request, and the response data passes verification.

Verification of the response data for instance 1:

a. The first UINT indicates the revision of the device's Connection Data Set, the value shall match class attribute 8 (Connection Data Set Revision).
b. For each Connection Structure, verify the data members as below:
   - The Connection Specifier shall match the one in the 0x4C request.
   - Verify Connection Application, Structure Format word per Vol 1 Table 5B-6.13 Connection Application Type, Status and Structure Format WORD Bit Definition.
   - According to the Format Bit 4 to parse and display the Connection Structure.
   - Verify Connection Flags per Vol 1 Table 5B-12.8.
   - (Not implemented yet) Initiate a connection (Exclusive Owner or Input Only) to the DUT according to the Connection Parameters specified in the Connection Structure. The connection shall be created successfully and produce I/O data.

6.3) Continue to send Connection Read request 0x4C with incremental Connection Specifier and Connection Structure Exclusion = 0 (Connection Name String should not be omitted) if the General Status is 0 and Additional Status word response is 0x0001 in the previous 0x4C.
Pass: The DUT shall respond success for the current 0x4C request and pass the verification of response data as described in 6.2).

When the General Status is 0 and Additional Status word response is 0x0000 in the previous 0x4C, send another Connection Read request 0x4C with incremental Connection Specifier.
Pass: The DUT shall return 0x16 (Object not exists).

4.48 Power Management Object Test x53

This section defines the Power Management Object conformance test. This test is required for all devices that implement the Power Management Object. This object can be implemented by any device.

Functional Description

This test verifies the:

1) Class attributes access rules and implementation rules for the Power Management Object.
2) Instance attributes access rules and implementation rules for the Power Management Object.
3) Common Service implementations for class and instance.
4) Object–specific and reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
6) Object behavior accessible from the network.

Procedure Definition

- Initialization
  The Power Management Object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access all the object supported attributes. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	**[Expected Values, Responses]**
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT (2), Attribute_Not_Supported]
Max_Instance (02)	[UINT, Attribute_Not_Supported]
Num_Instances (03)	[UINT, , Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT, UINT[size]), Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access all object supported attributes. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	**[Expected Values, Responses]**
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT, Service_Not_Supported(0x08)]
Max_Instance (02)	[UINT, Service_Not_Supported(0x08)]
Num_Instances (03)	[UINT, Service_Not_Supported(0x08)]
Optional_Attribute_List (04)	[UINT, Service_Not_Supported(0x08)]

Attribute Name (ID)	[Expected Values, Responses]
Optional_Service_List (05)	[UINT, Service_Not_Supported(0x08)]
Max_Class_Attribute_Id (06)	[UINT, Service_Not_Supported(0x08)]
Max_Instance_Attribute_Id (07)	[UINT, Service_Not_Supported(0x08)]
Connection Data Set Revision	[UINT, Service_Not_Supported(0x08)]
Capabilities	[DWORD, Service_Not_Supported(0x08)]

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance. If multiple Power Management Object instances exist in a device, the Conformance Test Tool will run Get Attribute Test, Set Attribute Test, and Common Service Test for each found instance.

2) Instance attributes access rules and implementation rules test.

- Request Get_Attribute_Single service (x0E) addressed to a valid instance to access all object supported attributes. Save the returned value for each Get_Attribute_Single request.

  If any instance attributes are supported, then all of the instance attributes shall be supported.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Power Management Command (01)	[DWORD]
Power Management Status (02)	[DWORD]
Client Path (03)	[Struct of UINT, EPATH]
Number of Power Management Modes (04)	[UINT]
Power Management Modes (05)	[Struct, Attribute_Not_Supported]
Sleeping State Support (06)	[BOOL]
Wake from Sleep Units (07)	[USINT, Attribute_Not_Supported]
Wake from Sleep Time (08)	UINT, Attribute_Not_Supported]

Request Set_Attribute_Single service (x10) addressed to a valid instance to access all supported attributes. Use the data that was previously returned by the corresponding Get_Attribute_Single service.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Power Management Command (01)	[Attribute_Not_Settable (x0E), Service_Not_Supported (x08)]
Power Management Status (02)	[Attribute_Not_Settable (x0E) , Service_Not_Supported (x08)]
Client Path (03)	[Attribute_Not_Settable (x0E) , Service_Not_Supported (x08)]
Number of Power Management Modes (04)	[Attribute_Not_Settable (x0E) , Service_Not_Supported (x08)]
Power Management Modes (05)	[Success, Attribute_Not_Supported, Service_Not_Supported (x08)]
Sleeping State Support (06)	[Attribute_Not_Settable (x0E) , Service_Not_Supported (x08)]
Wake from Sleep Units (07)	[Attribute_Not_Settable (x0E) , Service_Not_Supported (x08) , Attribute_Not_Supported]
Wake from Sleep Time (08)	[Attribute_Not_Settable (x0E) , Service_Not_Supported (x08) ,

Attribute Name (ID)	[Expected Values, Responses]
	Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.
  If Get_Attributes_All service is implemented, the response is also validatedby comparing the attribute value returned from this service to the values received from Get_Attribute_Single service. Attribute 1, 2, 3, 8, 9 values should match.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success, Service_not_Supported]
Get_Attribute_Single (14)	[Success_Response]
Reserved (all others between 0-49)	[Service_Not_Supported]

- Request each of the Common Services, 0 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success, Service_not_Supported]
Get_Attribute_Single (14)	[Success, Service_not_Supported]
Reserved (all others between 0-49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75, 77 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75, 77..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 – 99 (0x4B-0x63), addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each of the Reserved Services, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	**[Expected Responses]**
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

7) Behavior Tests

Power_Management service test: not totally implemented.

4.49 Power Curtailment Object Test x5C

This section defines the Power Curtailment Object conformance test. This test is required for all devices that implement the Power Curtailment Object. This object can be implemented by any device.

Functional Description

This test verifies the:

1) Class attributes access rules and implementation rules for the Power Curtailment Object.
2) Instance attributes access rules and implementation rules for the Power Curtailment Object.
3) Common Service implementations for class and instance.
4) Object–specific and reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
6) Object behavior accessible from the network.

Procedure Definition

- Initialization
  The Power Curtailment Object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 0.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access all the object supported attributes. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Attribute_Not_Supported, Service_Not_Supported(0x08)]
Max_Instance (02)	[UINT(1..65535) , Attribute_Not_Supported, Service_Not_Supported(0x08)]
Num_Instances (03)	[UINT(1..65535) , Attribute_Not_Supported, Service_Not_Supported(0x08)]
Optional_Attribute_List (04)	[(UINT, UINT[size]), Attribute_Not_Supported, Service_Not_Supported(0x08)]
Optional_Service_List (05)	[(UINT, UINT[size]), Attribute_Not_Supported, Service_Not_Supported(0x08)]
Max_Class_Attribute_Id (06)	[UINT (1..199), Attribute_Not_Supported, Service_Not_Supported(0x08)]
Max_Instance_Attribute_Id (07)	[UINT, Attribute_Not_Supported, Service_Not_Supported(0x08)]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access all object supported attributes. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Revision (01)	[Service_Not_Supported(0x08) , Attribute_Not_Supported]
Max_Instance (02)	[Service_Not_Supported(0x08) , Attribute_Not_Supported]
Num_Instances (03)	[Service_Not_Supported(0x08) , Attribute_Not_Supported]
Optional_Attribute_List (04)	[Service_Not_Supported(0x08) , Attribute_Not_Supported]
Optional_Service_List (05)	[Service_Not_Supported(0x08) , Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Max_Class_Attribute_Id (06)	[Service_Not_Supported(0x08) , Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[Service_Not_Supported(0x08) , Attribute_Not_Supported]

Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance. If multiple Power Curtailment Object instances exist in a device, the Conformance Test Tool will run Get Attribute Test, Set Attribute Test, and Common Service Test for each found instance.

2) Instance attributes access rules and implementation rules test.

- Request Get_Attribute_Single service (x0E) addressed to a valid instance to access all object supported attributes. Save the returned value for each Get_Attribute_Single request.

  If any instance attributes are supported, then all of the instance attributes shall be supported.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number of Curtailment Levels (01)	[UINT]
Curtailment Levels (02)	[STRUCT]
Maximum Number of Curtailment Levels (03)	[UINT]
Instance Capabilities (04)	[BYTE]
Instance Status (05)	[STRUCT]
Instance Options (06)	[BYTE]
Present Curtailment Level ID (07)	[UINT]
Present Expected Power (08)	[REAL]
Owner Path (09)	[STRUCT]
Uncurtailed Power (10)	[REAL]

Request Set_Attribute_Single service (x10) addressed to a valid instance to access all supported attributes. Use the data that was previously returned by the corresponding Get_Attribute_Single service.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Number of Curtailment Levels (01)	[Attribute_Not_Settable (x0E), Service_Not_Supported (x08)]
Curtailment Levels (02)	[Attribute_Not_Settable (x0E), Service_Not_Supported (x08)]
Maximum Number of Curtailment Levels (03)	[Attribute_Not_Settable (x0E), Service_Not_Supported (x08)]
Instance Capabilities (04)	[Attribute_Not_Settable (x0E), Service_Not_Supported (x08)]
Instance Status (05)	[Attribute_Not_Settable (x0E), Service_Not_Supported (x08)]
Instance Options (06)	[Attribute_Not_Settable (x0E), Service_Not_Supported (x08), Success]
Present Curtailment Level ID (07)	[Attribute_Not_Settable (x0E), Service_Not_Supported (x08)]
Present Expected Power (08)	[Attribute_Not_Settable (x0E), Service_Not_Supported (x08)]
Owner Path (09)	[Attribute_Not_Settable (x0E), Service_Not_Supported (x08)]
Uncurtailed Power (10)	[Attribute_Not_Settable (x0E), Service_Not_Supported (x08)]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.
  If Get_Attributes_All service is implemented, the response is also validatedby comparing the attribute value returned from this service to the values received from Get_Attribute_Single service. Attribute 1, 2, 3, 8, 9 values should match.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success, Service_not_Supported]
Get_Attribute_List (03)	[Success, Service_not_Supported]
Create (08)	[Success, Service_not_Supported]
Get_Attribute_Single (14)	[Success_Response, Service_not_Supported]
Reserved (all others between 0-49)	[Service_Not_Supported]

- Request each of the Common Services, 0 - 49, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
Get_Attributes_All (01)	[Success, Service_not_Supported]
Get_Attribute_List (03)	[Success, Service_not_Supported]
Set_Attribute_List (04)	[Success, Service_not_Supported]
Delete (09)	[Success, Service_not_Supported]
Get_Attribute_Single (14)	[Success]
Set_Attribute_Single (16)	[Success, Service_not_Supported]
Get_Member (24)	[Success, Service_not_Supported]
Reserved (all others between 0-49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75, 77 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75, 77..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 – 99 (0x4B-0x63), addressed to a valid Instance ID.

Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Establish_Ownership (x4B)	[Success]
Release_Ownership (x4C)	[Success]
Change_Ownership (x4D)	[Success]
Go_To_Level (x4E)	[Success]
Go_To_Level_Query (x4F)	[Success]
Capture_Level (x50)	[Success, Service_not_Supported]
Remove_Level (x51)	[Success, Service_not_Supported]
Revise_Level (x52)	[Success, Service_not_Supported]
Read_Level (x53)	[Success, Service_not_Supported]
Write_Level (x54)	[Success, Service_not_Supported]
Object–specific Codes (all others between 75..99)	[Service_Not_Supported (x08)]

- Request each of the Reserved Services, 100 - 255, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported (x08)]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests

Not implemented yet.

8) Behavior Tests

Not implemented yet.

5. Appendix A – CIP Object Test Definitions

This section is the template for use when defining new CIP Object Conformance tests.

Modify the existing sections by removing references to attributes and services that are not needed and adding the attribute and services that are needed. Complete the Error Tests and Behavior Tests sections as appropriate for the object you are defining. Refer to CIP Object Tests for a detailed discussion of the format and content of the CIP Object Conformance Test.

A.1 Example Object Test

This section defines the conformance test for the Example object. This test is required when the Example object is implemented in the device.

Functional Description

This test verifies the:

1) Class attributes access and implementation rules for the Example object.
2) Instance attributes access and implementation rules for the Example object.
3) Common Service implementations for class and instance.
4) Object–specific and reserved service implementations for class and instance.
5) Conformance when incorrect data is used to access attributes and services.
5.1) start list of Error Tests.
6) Object behavior accessible from the network.
6.1) start list of Behavior Tests.

Procedure Definition

- Initialization
  The Example object must be available. Log the object name and object test revision.
- Message Connection
  Establish an Explicit Messaging Connection. Set the EPR = 5000ms.

1) Class attribute access rules test.

- Request a Get_Attribute_Single service (x0E) addressed to the class, Instance ID zero, to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Revision (01)	[UINT (1), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance (02)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Num_Instances (03)	[UINT (1..65535), Service_Not_Supported, Attribute_Not_Supported]
Optional_Attribute_List (04)	[(UINT (1..199), UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Optional_Service_List (05)	[(UINT (1..99), UINT[size]), size is first UINT Service_Not_Supported, Attribute_Not_Supported]
Max_Class_Attribute_Id (06)	[UINT(6..199), Service_Not_Supported, Attribute_Not_Supported]
Max_Instance_Attribute_Id (07)	[UINT(0..199), Service_Not_Supported, Attribute_Not_Supported]

Attribute Name (ID)	[Expected Values, Responses]
Undefined (02..99)	[Service_Not_Supported, Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to the class, Instance ID zero, to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Any Attribute (00..99)	[Service_Not_Supported (x08)]

2) Instance attribute access rules test.
Note: Locate an instance of the object by requesting a Get_Attribute_Single, attribute 1, from instance 1 through first instance found. Any response other than Object_Does_Not_Exist (x16) indicates an instance is found. Use found instance ID for any tests that require an instance.

- Request a Get_Attribute_Single service (x0E) addressed to a valid instance to access attributes 00 - 99. Save the returned value for each Get_Attribute_Single request.
  Pass: The expected response from table below for each Get_Attribute_Single request.

Attribute Name (ID)	[Expected Values, Responses]
Undefined (00)	[Attribute_Not_Supported (x14)]
Attribute_1 (01)	[expected values, or Error_Responses]
Attribute_2 (02)	[expected values, or Error_Responses]
Undefined (03..99)	[Attribute_Not_Supported]

- Request a Set_Attribute_Single service (x10) addressed to a valid instance to access attributes 00 - 99. Use data returned by the appropriate Get_Attribute_Single service.
  Pass: The expected response from table below for each Set_Attribute_Single request.

Attribute Name (ID)	[Expected Responses]
Undefined (00)	[Service_Not_Supported (x08), Attribute_Not_Supported (x14)]
Attribute_1 (01)	[Service_Not_Supported, Attribute_Not_Settable (x0E)]
Attribute_2 (02)	[Success_Response, Service_Not_Supported, Attribute_Not_Settable]
Undefined (03..99)	[Service_Not_Supported, Attribute_Not_Supported]

3) Common Services Test

- Request each of the Common Services 00 - 49 addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Common Service request.

Service Name (Code)	[Expected Responses]
Reserved (00)	[Service_Not_Supported (x08)]
(01..13)	[Service_Not_Supported]
Get_Attribute_Single (14)	[requested value] [Service_Not_Supported] if no attributes are supported
Reserved (15..49)	[Service_Not_Supported]

- Request each of the Common Services, 00 - 49, addressed to a valid Instance ID.
  Pass: Expected responses for Common Services addressed to the instance level object

Service Name (code)	[Expected Responses]
(00..13)	[Service_Not_Supported (x08)]
Get_Attribute_Single (14)	[requested value]
Reserved (15)	[Service_Not_Supported]
Set_Attribute_Single (16)	[Success_Response, Attribute_Not_Settable (x0E)]
Reserved (17..49)	[Service_Not_Supported]

4) Object–specific and Reserved Services Test

- Request each Object–specific Service, 75 - 99, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the class, Instance ID zero.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

- Request each Object–specific Service, 75 - 99, addressed to a valid Instance ID.
  Pass: The expected response from table below for each Object–specific Service request.

Service Name (Code)	[Expected Responses]
Object–specific Codes (75..99)	[Service_Not_Supported (x08)]

- Request each Reserved Service, 100 - 255, addressed to the a valid Instance ID.
  Pass: The expected response from table below for each Reserved Service request.

Service Name (Code)	[Expected Responses]
Reserved Codes (100..127)	[Service_Not_Supported]
Reserved Codes (128..255)	[Service_Not_Supported, Invalid_Parameter (x20), No_Response]

5) Error Tests.

5.1)

- Request a service
  Pass: The expected response

6) Behavior tests.

6.1)

- Request a service
  Pass: The expected response

Delete the connections at the conclusion of this test and leave the device in the On–Line state.