EPRI | ELECTRIC POWER RESEARCH INSTITUTE

Common Information Model Meter Outage Nominal Interoperability Test Procedure

EPRI CIM Interoperability Test Procedure (ECITP) 2.05

2011 TECHNICAL REPORT

Common Information Model Meter Outage Nominal Interoperability Test Procedure

EPRI CIM Interoperability Test Procedure (ECITP) 2.05

EPRI Project Manager
J. Simmins

ELECTRIC POWER RESEARCH INSTITUTE

3420 Hillview Avenue
Palo Alto, CA 94304-1338
USA

PO Box 10412
Palo Alto, CA 94303-0813
USA

800.313.3774
650.855.2121

askepri@epri.com

www.epri.com

1024445
Final Report, December 2011

DISCLAIMER OF WARRANTIES AND LIMITATION OF LIABILITIES

THE FOLLOWING ORGANIZATION(S), UNDER CONTRACT TO EPRI, PREPARED THIS REPORT:

EnerNex

NOTE

For further information about EPRI, call the EPRI Customer Assistance Center at 800.313.3774 or e-mail askepri@epri.com.

Acknowledgments

The following organizations, under contract to the Electric Power Research Institute (EPRI), prepared this report:

EnerNex
620 Mabry Hood Road
Suite 300
Knoxville, TN 37932

Principal Investigator
K. Stefferud
B. Muschlitz

Boreas Group
730 S. Elizabeth St.
Denver, Colorado 80209

Principal Investigator
R. Sarfi
R. Boswell

This report describes research sponsored by EPRI. EPRI would like to acknowledge the support of the following organizations:

Consumer's Energy
American Electric Power
Southern California Edison

This publication is a corporate document that should be cited in the literature in the following manner:

Common Information Model Meter Outage Nominal Interoperability Test Procedure: EPRI CIM Interoperability Test Procedure (ECITP) 2.05. EPRI, Palo Alto, CA: 2011. 1024445.

Abstract

The CIM Meter Outage Nominal Interoperability Test Procedure is one in a series of EPRI CIM Interoperability Test Procedures (ETIPs) created by EPRI whose purpose is to thoroughly document the actors, interfaces, and test steps for the interoperability testing of specific parts of the International Electrotechnical Commission (IEC) Common Information Model (CIM) standard. The Test Procedures are initially being used for EPRI demonstration tests and are intended, over time, to form the basis of a set of CIM test procedures to be used for:

- Testing labs set up under a formal organizational framework for Smart Grid testing and certification, such as that called for by the Smart Grid Testing & Certification Committee (SGTCC) in its Interoperability Process Reference Manual (IPRM) whose goal is the formalized validation of interoperability of combinations of vendor products
- Organizations sponsoring routine standards-validation CIM interoperability testing

The initial series of Test Procedures covers several message sets related to the IEC 61868-9 (Meter Reading & Control) standard. This document covers meter outage nominal test procedures.

Keywords
Smart meters
Common Information Model (CIM)
Interoperability Test Procedures
Smart Grid Testing and Certification Committee (SGTCC)

Table of Contents

List of Figures

List of Tables

Section 1: Introduction

EPRI initiatives have provided important research contributing to bodies of work that have become International Electrotechnical Commission (IEC) interface standards—including the Common Information Model (CIM), Inter-Control Center Communications Protocol (ICCP) and 61850. These standards provide the basis for model-driven information exchange within and between control centers, substations, and other systems supporting utility operations. The implementation of a smarter grid is facilitated by the development of interoperability standards such as these. Having a universally accepted, well defined, model-driven information exchange strategy is the most realistic and cost effective way to implement a Smart Grid.

The initial CIM standard addressed interoperatbility between Energy Management Systems and other control center applications. The primary challenge in the future is to extend CIM beyond the control center and prove that it is stable and fully implementable. Once CIM is extended, it is expected to allow full data management and exchange between the transmission, distribution, planning, and generation areas of the enterprise and with outside entities. When new data provided by these systems can be accessed, utilities will have a greatly improved chance of achieving efficiencies that will lower the cost of future system upgrades and integration.

One measure of stability is having standard set of test cases that can be used to repeatably and reliably test the repeatably test the interoperability of two systems. This document is one in a series of EPRI-authored documents that provide the detailed information (steps, instructions, and interface definitions) related to such test cases.

Background

The EPRI CIM Interoperability Test Procedures are designed to support interoperability testing of the IEC CIM standards (61968, 61970, and 62325). The initial collection of Test Procecures focuses on 61968-9 (Meter Reading & Control). The scope of 61968-9 Test Procedures is shown in Figure 2-1This document, CIM Meter Outage Nominal Interoperability Test Procedure, comprises one of component procedures of the 61968-9 collection of Test Procedures.

Purpose of This Document

The purpose of the EPRI CIM test procedures is to thoroughly describe how system-to-system interoperability testing using IEC standard CIM messages can be performed. The test procedures are intended to be utilized by testing labs set up under a formal organizational framework for Smart Grid testing and certification, such as that called for by the Smart Grid Testing & Certification Committee (SGTCC) in its Interoperabiltiy Process Reference Manual (IPRM). In this use, the EPRI CIM test procedures address the need for a more formalized validation of interoperability of combinations of vendor products. The test procedures are also intended to be utilized by organizations sponsoring routine standards-validation CIM interoperability testing, as has historically been done by EPRI, UCA International, EdF, and ENTSO-E.

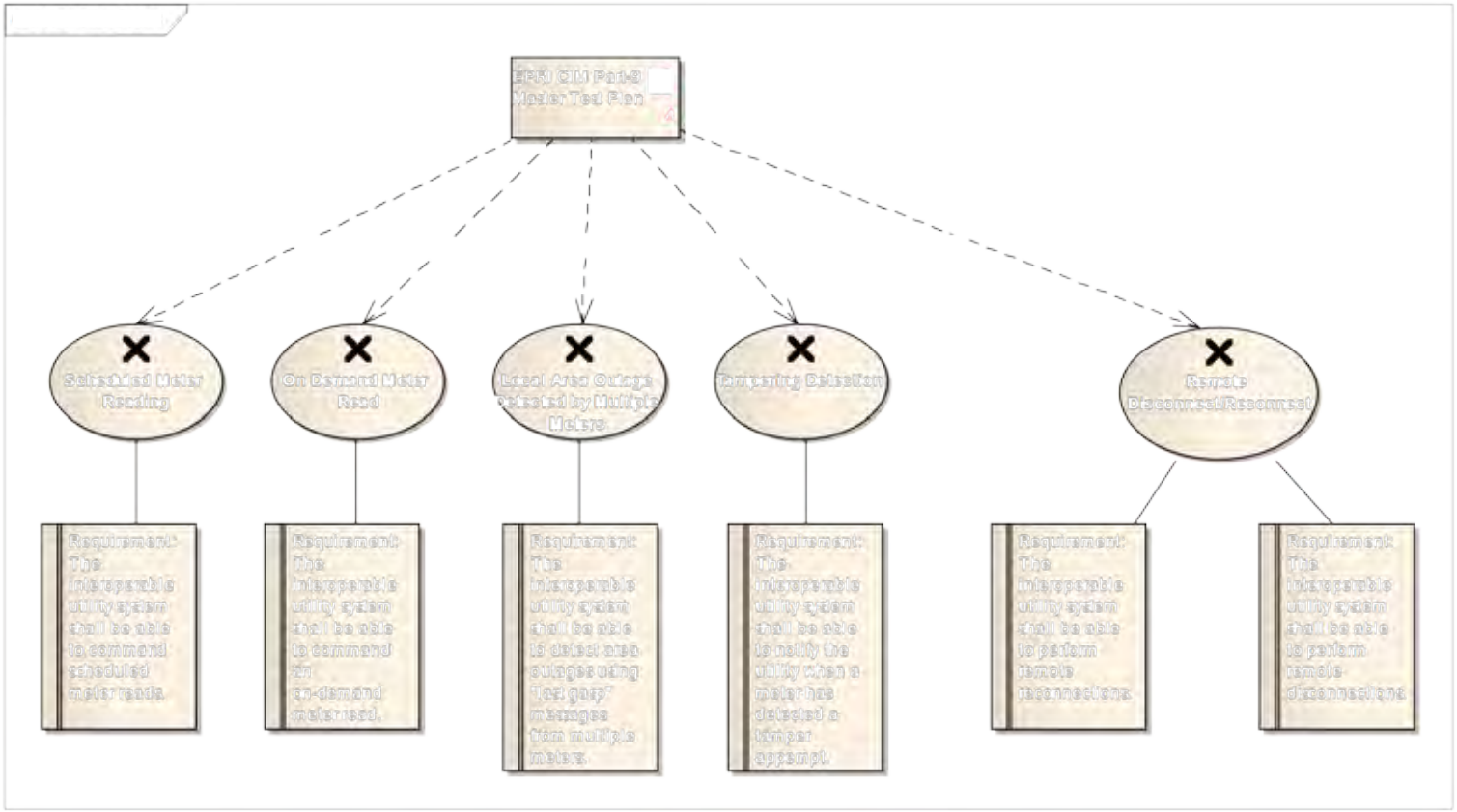

Figure 1-1
The Scope of the EPRI CIM 61968-9 Test Plans

Contents of This Document

Chapter 2 contains the references used in the preparation of this test procedure.

Chapter 3 contains a description of the test set up and what is to be tested.

Chapter 4 has the step by step testing procedure.

Appendix A contains the technical information for setting up a test.

Appendix B contains the XSDs and WSDLs used in the test procedure.

Section 2: References

Normative

IEC 61968-9 Interfaces for Meter Reading and Control, First Edition, International Electrotechnical Commission, Geneva Switzerland.

Informative

The business requirements are derived from the requirements published by Southern California Edison and those at smartgridipedia.org.

Use cases from Southern California Edison:

http://www.sce.com/CustomerService/smartconnect/industry-resource-center/use-cases.htm?from=usecases

- Use Case D4 ver. 1.2050106 (v 1.5 with comments) REQ0001 - AMI System shall detect and report outages.
- Use Case D4 ver. 1.2050106 REQ0010 - Meter shall report meter ID and time for outage and restoration reports.
- Use Case D4 ver. 1.2050106 REQ0011 - Meter shall communicate that its power has been restored when it recovers from an outage.
- Use Case D4 ver. 1.2050106 REQ0012 - AMI System shall record outage information on a meter by meter basis.
- Use Case D4 ver. 1.2050106 REQ0013 - AMI System shall record the duration of outage for later statistical analysis.

Use cases from smartgridipedia:

http://www.smartgridipedia.org/index.php/D4_-_Scenario_1

- Use Case D4 AMI-ENT version REQ0027 - AMI System shall report restoration of 99% of all affected meters within 30 minutes (GOAL: before field crew leaves) of restoration to the Outage Management System
- Use Case D4 AMI-ENT version REQ-D4002 - Send AMI Outage Events v1.2 -. See http://www.smartgridipedia.org/index.php/D4_-_Scenario_1
- Use Case D4 AMI-ENT version REQ-D4003 - Show Updated AMI Device Event.

- Use Case D4 AMI-ENT version REQ-D4005 - Publish Outage Record.
- Use Case D4 AMI-ENT version REQ-D4006 - Update Outage Record
- Use Case D4 AMI-ENT version REQ-D4007 - Send meter service order request from OMS to AMI Management System.

Section 3: Test Overview

Meter Outage Nominal Interoperability Test

This Meter Outage Nominal Interoperability Test verifies that the Outage Management System (OMS) and related systems can report outages based on outage detection performed by smart meter systems. Units Under Test (UUTs) are shown in Figure 3-1. This type of diagram is adapted from X.291 specification of the International Telecomunications Union[1] and is the reference diagram for this test. The nomenclature for the systems mentioned in this test procedure is taken from the IEC 61968-1 standard.[2] The relationship between the common names for the application and their functional names, found in the 61968 document are given below in Table 3-1.

Table 3-1
Comparison of Common Application Names and Function Names for Units Under Test for the Meter Outage Nominal Interoperability Test

Application Name	IEC 61968-1 Function Name
Advanced Meter Infrastructure (AMI) Head End	MR – RMR (Meter Reading and Control – Meter Reading)
Meter Data Management System (MDMS)	MR – MOP (Meter Reading and Control – Meter Operations)
Outage Management System (OMS)	NO – FLT (Network Operations – Fault Management)

Last gasp messages, indicating that the smart meter no longer has power, are sent to the MR-RMR or AMI (Advanced Metering Infrastructure) Head End via a proprietary protocol and. These messages are proprietary and are out of the scope of this test. The last gasp can generate an end device event in a CIM compliant AMI application and can be sent to a Meter Data Management System (MDMS) or an Outage Management System (OMS) or NO-FLT (Network Operations – Fault Management) using the 61968-1 nomenclature.

Actual meters, if available, may be used in the test system.

---

[1] TTU-T X291 Specification, International Telecommunications Union, 1996.

[2] IEC 61968-1 Interface architecture and general requirements, International Electrotechnical Commission, Geneva, Switzerland.

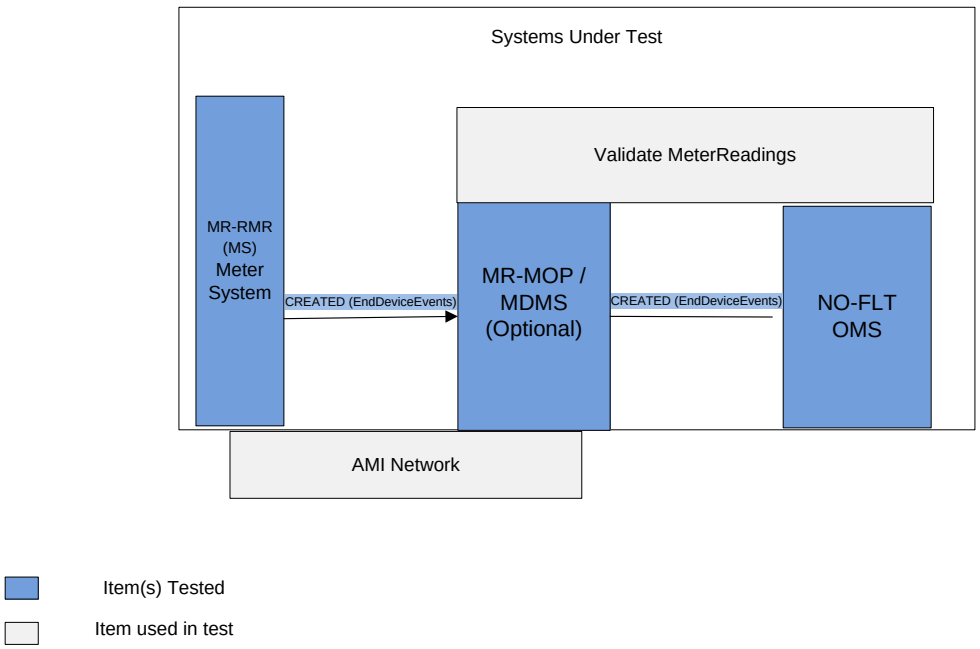

Figure 3-1
Units Under Test (UUT) for the Meter Outage Nominal Interoperability Test

Interface Description

The system interfaces are shown in Figure 3-1 ECITP 2.05 Meter Message Sequence Flow Diagram. As shown in the diagram, systems under test are the OMS, MDMS and AMI Head End. The CIM message used is EndDeviceEvents with EndDeviceEventType of 3.26.0.85 for the last gasp power outage messages and 3.26.0.216 for the power restoration event. These End Device Events as well as the complete set of End Device Events with their meanings can be found in the IEC 61968-9 Annex E, titled EndDeviceEventType Enumerations.

As shown in Figure 3-1 ECITP 2.05 below, the AMI Head End (MS) receives the power out notifications from individual smart meters in proprietary non-standard format as is the case in implementations to date. The AMI Head End (MS) and MDMS (MR-MOP) systems perform outage detection and outage validation prior to creating the outage events (a type of end device events) using standard CIM format. CIM formatted meter outage events are then forwarded to the OMS (NO-FLT). Meters can be de-energized for reasons such as bill nonpayment and maintenance purposes (which do not constitute power outage events) and currently, at least some smart meters are reporting false positive meter outage events. Therefore filtering and validation of the meter power out events is assumed to be performed by the AMI Head End (MS) or MDMS (MR-MOP) systems such that all issued EndDeviceEvents can be assumed to be true reflections of power outage conditions.

Ideally, the AMI head end would use an IEC 61968-9 message for the notification of the MDMS that an outage event has occurred. This message would be sent in a publish/subscribe exchange (Figure 3-2) and would allow the propagation of the outage event directly to an OMS (Figure 3-3).

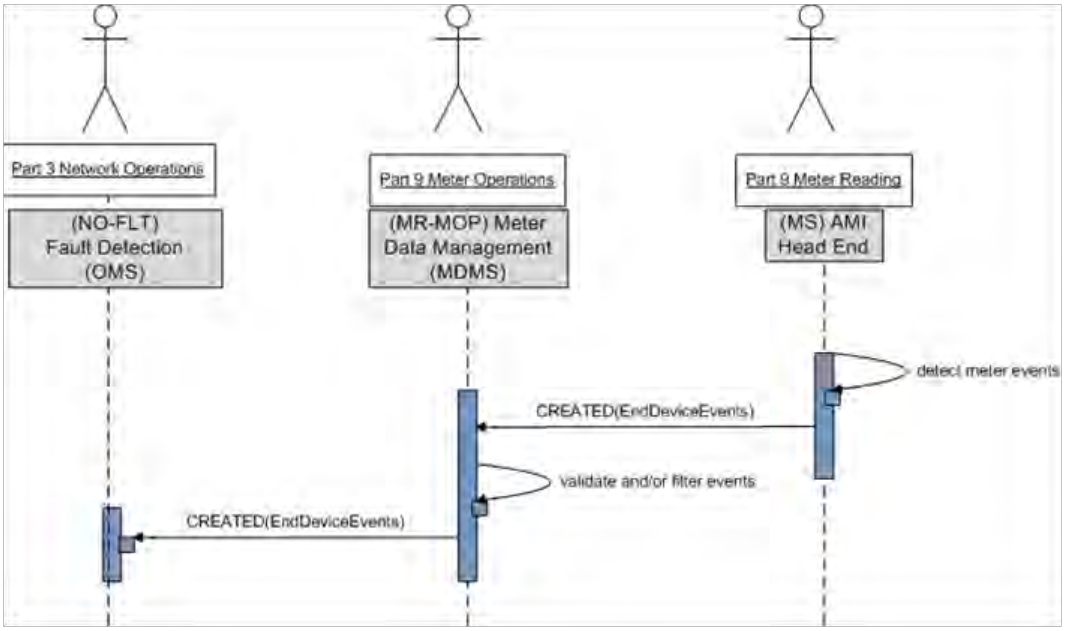

Figure 3-2
ECITP 2.05 Meter Message Sequence Flow Diagram with IEC 61968-9 Messages from the MS (AMI Head End) to the MR-MOP (Meter Data Management System) in a Publish/Suscribe Exchange[3]

[3] IEC 61968-9 Second Edition, International Electrotechnical Commission, Geneva Switzerland, 2011 (in Committee Draft)

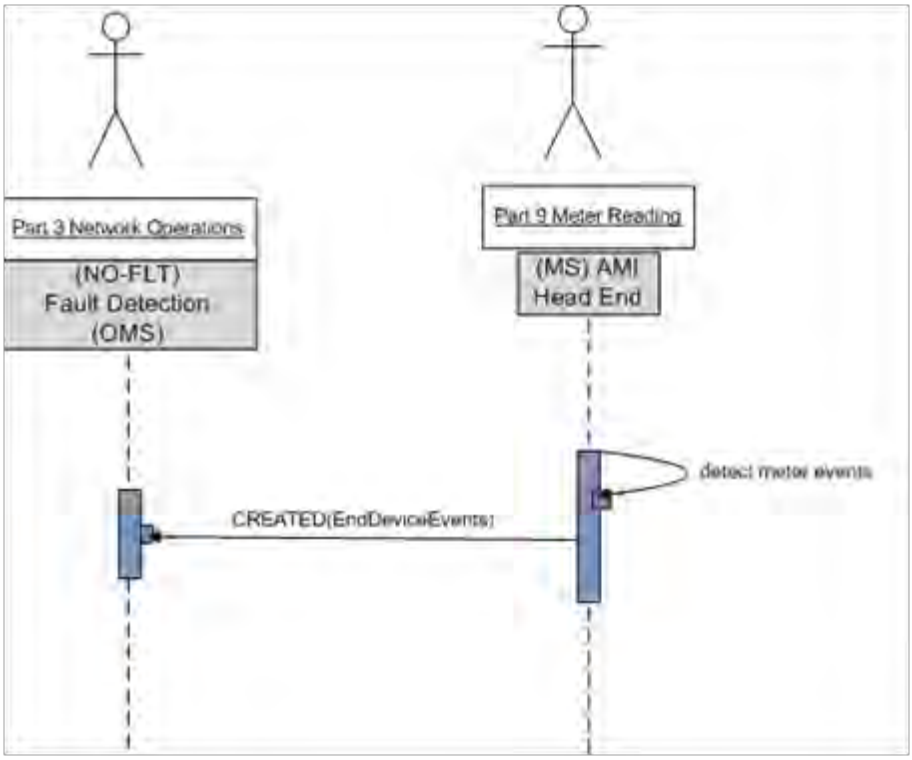

Figure 3-3
ECITP 2.05 Meter Message Sequence Flow Diagram with IEC 61968-9 Messages from the Meter System (AMI Head End) to the Outage Management System in a Publish/Suscribe Exchange

Test Description

The ability of the MS (AMI Head End) and an OMS (Outage Management System) to detect and report electric power failures/outages is validated for the systems under test. Additionally, the presence of a MR-MOP (MDMS) and an intermediary system where additional functions are peformed.is also tested. Standard IEC 61968-9 meter messages to/from the Meter System (MS) and from/to the OMS are used to report an outage for first an individual and then multiple meters. Single point of failure and nested outages where multiple failures have caused power outages are simulated. The systems' response to the simulated failures is validated. Note that IEC 61968-9 does not currently mandate use of any particular error codes, but does provide sample error codes

The IEC 61968-9 generic transmission of meter messages is shown above in Figure 3-1 in Universal Modeling Language (UML) format.

The main test scenarios are as follows:

1. 1. A simulated power failure occurs. The MS (AMI Head End) detects and validates the failure and then sends an EndDeviceEvents message to the OMS. The OMS is monitored to verify that the power failure is detected and an outage is declared.
2. 2. The message sequence for the EndDeviceEvent message is shown in Figure ECITP 2.-05 -3 in Universal Modeling Language (UML) format. The required IEC 61968 part 9 eXtensible Markup Language (XML) Schema Definition (XSD) is used to send the EndDeviceEvent message.The simulated power failure ends. The Meter System detects and validates the power restoration and then sends an EndDeviceEvents message to the OMS. The OMS is monitored to verify that the power restoration is detected and the outage is ended.
3. 3. The meter simulator (or actual meters, or a test harness) is used to inject multiple nested power outages. The Meter System detects and validates the failures and then sends one or more than one EndDeviceEvents messages to the OMS. The OMS is monitored to verify that the multiple meter failures are detected and an area outage is declared. The simulated power outage is partially corrected with some area(s) still out. The OMS is monitored to verify that it clears the outage reports for the simulated area with restored power, and that the multiple meter failures still existing are detected and reported to the operators.

The simulated power outage is completely restored. The OMS is monitored to verify that it clears the outage reports for the the entire area.
In all cases, the required IEC 61968-9 eXtensible Markup Language (XML) Schema Definition (XSD) is used to send the EndDeviceEvents message(s). The EndDeviceEvents message format is defined in IEC Standard 61968-9 and the XML XSD definition is contained in that document's annex A.

See Appendix B for a sample of XML format and content of the meter power outage and power outage restoration EndDeviceEvents messages used for this test. These XML document are to be considered informative in nature, providing examples of the XSDs andWSDLs that would be generated for the actual test.

Section 4: Test Steps and Results

For each Unit under Test (UUT) detailed test procedure steps specific to that role are detailed.

Table 4-1
Test Steps for MDM

Step #	Ref. #	Source Component	Destination Component	Detailed Step	Pass/Fail
1	Use Case D4 AMI-ENT version REQ0001	AMI Meter/ simulated meter	AMI Meter/ simulated meter	Force a power outage for at least one meter.	
2		AMI Meter/ simulated meter	MDMS	MDMS receives the power outage message for the meter /meter(s) with a forced power outage.	Pass/Fail
3	Use Case D4 ver. 1.2050106 REQ0010	MDM	OMS	MDMS forwards and combines (if there is more than one outage) outage messages. Verify the MDM logs the power outage event for the affected meter(s) and correctly lists the meter ID and outage time in the MDMS event log.	Pass/Fail
4		AMI Meter/ simulated meter	AMI Meter/ simulated meter	Restore power to the affected meter(s).	
5	Use Case D4 AMI-ENT version REQ-D4003	AMI Meter/ simulated meter	MDMS	MDMS receives the power restoration message from the meter /meter.	Pass/Fail

Table 4-1 (continued)
Test Steps for MDM

Step #	Ref. #	Source Component	Destination Component	Detailed Step	Pass/Fail
6	Use Case D4 ver. 1.2050106 REQ0011, Use Case D4 ver. 1.2050106 REQ0013	MDMS	OMS	MDMS forwards and combines (if there is more than one outage) outage restoration messages. Verify the MDM logs the power restoration event for the affected meter and correctly lists the total outage time in the MDMS event log. A Pass occurs if the outage time appears in the log.	Pass/Fail
7	Use Case D4 ver. 1.2050106 REQ0001	AMI Meter/ simulated meter	AMI Meter/ simulated meter	Force a power outage for 200 meters with multiple nested faults. Record the time of the outage.	
8	ARCH - D4-USE CASE v1.5 with comments.doc, dated 04/19/06 NF requirement #4	MDM	OMS	MDMS forwards and combines (if there is more than one outage) outage messages for the meters with a forced power outage for the multiple fault area. Record the time that outages are detected by the MDMS. Record the time the MDMS reports the outage for all meters. Verify the MDMS detects end device events (outages) for all 200 meters within 5 minutes	Pass/Fail
10		AMI Meter/ simulated meter	MDMS	Restore power to the some, but not all affected meters in the nested configuration.	
11	Use Case D4 AMI-ENT version REQ-D4003	AMI Meter/ simulated meter	MDMS	Verify the MDM sends power restoration event messages for the correct set of meters. Verify the meters with no power are still indicated as out by the MDMS.	Pass/Fail

Table 4-1 (continued)
Test Steps for MDM

Step #	Ref. #	Source Component	Destination Component	Detailed Step	Pass/Fail
12	Use Case D4 ver. 1.2050106 REQ0011, Use Case D4 ver. 1.2050106 REQ0013	AMI Meter/ simulated meter	MDMS	Verify the MDM logs the power outage event for each affected meter and correctly lists the total outage time in the log for each meter.	Pass/Fail
13		AMI Meter/ simulated meter		Restore power to the remaining meters.	
14	Use Case D4 ver. 1.5 AMI-ENT version REQ0027 -	AMI Meter/ simulated meter	MDMS	Verify the MDM sends power restoration event messages for the correct set of meters within 30 minutes	Pass/Fail
15		AMI Meter/ simulated meter	MDMS	Verify the MDMS logged the power outage event for each affected meter(s) and correctly listed the total outage time in the log for each meter.	Pass/Fail

Table 4-2
Test Steps for OMS

Step #	Ref. #	Source Component	Destination Component	Detailed Step	Pass/Fail
1	Use Case D4 AMI-ENT version REQ0001	AMI Meter/ simulated meter	AMI Meter/ simulated meter	Force a power outage for at least one meter.	
2	IEC 61968 version 10v30, EndDeviceEvents XSD	AMI Meter/ simulated meter	OMS	OMS indicates a power outage for the meter(s) with a forced power outage.	Pass/Fail
4		AMI Meter/ simulated meter	AMI Meter/ simulated meter	Restore power to the affected meter(s).	
5	Use Case D4 AMI-ENT version REQ-D4003	AMI Meter/ simulated meter	OMS	Verify the OMS indicates that the power outage has been cleared.	Pass/Fail
7	Use Case D4 ver. 1.2050106 REQ0001	AMI Meter/ simulated meter	AMI Meter/ simulated meter	Force a power outage for 200 meters with multiple nested faults. Record the time of the outage.	
8	ARCH - D4-USE CASE v1.5 with comments.doc, dated 04/19/06 NF requirement #4	AMI Meter/ simulated meter	OMS	OMS indicates a power outage for the meter(s) with a forced power outage for the multiple fault area. Record the time that outages are detected by the OMS. Record the time the OMS reports the outage for all meters. Verify the OMS detects errors for all 200 meters within 5 minutes	Pass/Fail
10		AMI Meter/ simulated meter		Restore power to the some, but not all affected meter(s). Restore power in only some meters in a nested configuration.	

Table 4-2 (continued)
Test Steps for OMS

Step #	Ref. #	Source Component	Destination Component	Detailed Step	Pass/Fail
11	Use Case D4 AMI-ENT version REQ-D4003	AMI Meter/ simulated meter	OMS	Verify the OMS indicates that the power outage has been cleared for the correct set of meters. Verify the meters with no power are still indicated as out.	Pass/Fail
13		AMI Meter/ simulated meter		Restore power to the remaining meters.	
14	Use Case D4 ver. 1.5 AMI-ENT version REQ0027 -	AMI Meter/ simulated meter	OMS	Verify the OMS indicates that the power outage has been cleared for the correct set of meters within 30 minutes. Verify all meters are now indicated as having power.	Pass/Fail

Table 4-3
Test Steps for AMI

Step #	Ref. #	Source Component	Destination Component	Detailed Step	Pass/Fail
1	Use Case D4 AMI-ENT version REQ0001	AMI Meter/ simulated meter	AMI Meter/ simulated meter	Force a power outage for at least one meter.	
2	IEC 61968 version 10v30, EndDeviceEvents XSD	AMI Meter/ simulated meter	AMI Head End	AMI Head End indicates a power outage for the meter(s) with a forced power outage.	Pass/Fail
3	Use Case D4 ver. 1.2050106 REQ0010	AMI Head End		Verify the AMI Head End logs the power outage event for the affected meter(s) and correctly lists the meter ID and outage time in the log.	Pass/Fail
4		AMI Meter/ simulated meter	AMI Meter/ simulated meter	Restore power to the affected meter(s).	
5	Use Case D4 AMI-ENT version REQ-D4003	AMI Meter/ simulated meter	AMI Head End	Verify the AMI Head End indicates that the power outage has been cleared.	Pass/Fail
6	Use Case D4 ver. 1.2050106 REQ0011, Use Case D4 ver. 1.2050106 REQ0013	AMI Meter/ simulated meter	AMI Head End	Verify the AMI Head End logs the power outage event for the affected meter(s) and correctly lists the total outage time in the log.	Pass/Fail
7	Use Case D4 ver. 1.2050106 REQ0001	AMI Meter/ simulated meter	AMI Meter/ simulated meter	Force a power outage for 200 meters with multiple nested faults. Record the time of the outage.	

Table 4-3 (continued)
Test Steps for AMI

Step #	Ref. #	Source Component	Destination Component	Detailed Step	Pass/Fail
8	ARCH - D4-USE CASE v1.5 with comments.doc, dated 04/19/06 NF requirement #4	AMI Meter/ simulated meter	AMI Head End	AMI Head End indicates a power outage for the meter(s) with a forced power outage for the multiple fault area. Record the time that outages are detected by the AMI Head End. Record the time the AMI Head End reports the outage for all meters. Verify the AMI Head End detects errors for 99% of the 200 meters within 5 minutes.	Pass/Fail
9	Use Case D4 ver. 1.2050106 REQ0010	AMI Head End		Verify the AMI Head End logs the power outage event for the affected meter(s) and correctly lists the meter ID and outage time in the log for each affected meter.	Pass/Fail
10		AMI Meter/ simulated meter		Restore power to the some, but not all affected meter(s). Restore power in only some meters in a nested configuration.	
11	Use Case D4 AMI-ENT version REQ-D4003	AMI Meter/ simulated meter	AMI Head End	Verify the AMI Head End indicates that the power outage has been cleared for the correct set of meters. Verify the meters with no power are still indicated as out.	Pass/Fail
12	Use Case D4 ver. 1.2050106 REQ0011, Use Case D4 ver. 1.2050106 REQ0013	AMI Meter/ simulated meter	AMI Head End	Verify the AMI Head End logs the power outage event for each affected meter(s) and correctly lists the total outage time in the log for each meter.	Pass/Fail

Table 4-3 (continued)
Test Steps for AMI

Step #	Ref. #	Source Component	Destination Component	Detailed Step	Pass/Fail
13		AMI Meter/ simulated meter		Restore power to the remaining meters.	
14	Use Case D4 ver. 1.5 AMI-ENT version REQ0027 -	AMI Meter/ simulated meter	AMI Head End	Verify the AMI Head End indicates that the power outage has been cleared for 99% of the correct set of meters within 30 minutes.	Pass/Fail
15		AMI Meter/ simulated meter	AMI Head End	Verify the AMI Head End logged the power outage event for each affected meter(s) and correctly listed the total outage time in the log for each meter.	Pass/Fail

Appendix A: Test Technical Information

Test Set Up

The test harness is designed to run on a cloud environment. The cloud environment used in this project is the Amazon EC2 cloud environment. Each vendor would have their own Amazon EC2 instance to run their copy of the test harness. There are several steps required to create a new instance, but each step is straightforward.Amazon EC2 is a cloud environment that makes it simple to create, manage, and deploy web based applications. It's an excellent choice for hosting the test harness, as it is easy to generate new instances on demand. Also, the service itself is extremely reliable. The cloud is an affordable option. The test harness has a very small footprint, which means that the lowest cost services are sufficient.

The details of establishing a test instance may be found in Appendix A – EPRI Interoperability Test Harness How-To-Deploy Tutorial in *CIM Conformity and Interoperability Test Procedure Development.*EPRI, Palo Alto, CA: 2011. 1024450.

Test Invocation

The details of establishing a test instance may be found in Appendix B – EPRI Test Harness Tutorial in *CIM Conformity and Interoperability Test Procedure Development.*EPRI, Palo Alto, CA: 2011. 1024450.

Schedule Test

1. Coordinate test scheduling with John Simmins at jsimmins@epri.com**.**

Execute Test

1. Start system(s) under test.
2. Verify connectivity of systems
3. Execute test steps as defined in section 5 for each system under test.
4. At test conclusion, receive and review recorded test status information.

Test XSDs

This test uses the standard 61968-1 message header and EndDeviceEvent payload, included in Appendix A.

Test Results

MDMS, Head End and OMS tests pass if each test step with a pass/fail in the far right column passes.

Appendix B: Detailed Description of Each Interface

Detailed descriptions of each interface, keyed to interface numbers from diagram.

WSDLs

(Created)EndDeviceEvents

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions name="SendEndDeviceEvents" xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:wsi="http://ws-
i.org/schemas/conformanceClaim/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
targetNamespace="http://iec.ch/TC57/2010/EndDeviceEvents"
xmlns:tns="http://iec.ch/TC57/2010/EndDeviceEvents"
xmlns:infoMessage="http://www.iec.ch/TC57/2010/EndDeviceEventsMessage">
 <wsdl:types>
 <xs:schema targetNamespace="http://www.iec.ch/TC57/2010/EndDeviceEventsMessage"
elementFormDefault="qualified">
  <xs:include schemaLocation="xsd/EndDeviceEventsMessage.xsd" />
  </xs:schema>
  </wsdl:types>
- <!--
 Message Definitions
  -->
<wsdl:message name="CreatedEndDeviceEventsEventMessage">
  <wsdl:part name="CreatedEndDeviceEventsEventMessage"
element="infoMessage:CreatedEndDeviceEvents" />
  </wsdl:message>
<wsdl:message name="ChangedEndDeviceEventsEventMessage">
  <wsdl:part name="ChangedEndDeviceEventsEventMessage"
element="infoMessage:ChangedEndDeviceEvents" />
  </wsdl:message>
```

```
<wsdl:message name="ClosedEndDeviceEventsEventMessage">
  <wsdl:part name="ClosedEndDeviceEventsEventMessage"
     element="infoMessage:ClosedEndDeviceEvents" />
     </wsdl:message>
<wsdl:message name="CanceledEndDeviceEventsEventMessage">
  <wsdl:part name="CanceledEndDeviceEventsEventMessage"
     element="infoMessage:CanceledEndDeviceEvents" />
     </wsdl:message>
 <wsdl:message name="DeletedEndDeviceEventsEventMessage">
  <wsdl:part name="DeletedEndDeviceEventsEventMessage"
     element="infoMessage:DeletedEndDeviceEvents" />
     </wsdl:message>
 <wsdl:message name="ResponseMessage">
  <wsdl:part name="ResponseMessage" element="infoMessage:EndDeviceEventsResponseMessage" />
     </wsdl:message>
 <wsdl:message name="FaultMessage">
  <wsdl:part name="FaultMessage" element="infoMessage:EndDeviceEventsFaultMessage" />
     </wsdl:message>
  - <!--
   Port Definitions
     -->
 <wsdl:portType name="EndDeviceEvents_Port">
 <wsdl:operation name="CreatedEndDeviceEvents">
  <wsdl:input name="CreatedEndDeviceEventsRequest"
     message="tns:CreatedEndDeviceEventsEventMessage" />
  <wsdl:output name="CreatedEndDeviceEventsResponse" message="tns:ResponseMessage" />
  <wsdl:fault name="CreatedEndDeviceEventsFault" message="tns:FaultMessage" />
     </wsdl:operation>
 <wsdl:operation name="ChangedEndDeviceEvents">
  <wsdl:input name="ChangedEndDeviceEventsRequest"
     message="tns:ChangedEndDeviceEventsEventMessage" />
  <wsdl:output name="ChangedEndDeviceEventsResponse" message="tns:ResponseMessage" />
  <wsdl:fault name="ChangedEndDeviceEventsFault" message="tns:FaultMessage" />
     </wsdl:operation>
 <wsdl:operation name="CanceledEndDeviceEvents">
  <wsdl:input name="CanceledEndDeviceEventsRequest"
     message="tns:CanceledEndDeviceEventsEventMessage" />
  <wsdl:output name="CanceledEndDeviceEventsResponse" message="tns:ResponseMessage" />
  <wsdl:fault name="CanceledEndDeviceEventsFault" message="tns:FaultMessage" />
     </wsdl:operation>
  <wsdl:operation name="ClosedEndDeviceEvents">
  <wsdl:input name="ClosedEndDeviceEventsRequest"
     message="tns:ClosedEndDeviceEventsEventMessage" />
  <wsdl:output name="ClosedEndDeviceEventsResponse" message="tns:ResponseMessage" />
  <wsdl:fault name="ClosedEndDeviceEventsFault" message="tns:FaultMessage" />
     </wsdl:operation>
- <wsdl:operation name="DeletedEndDeviceEvents">
  <wsdl:input name="DeletedEndDeviceEventsRequest"
     message="tns:DeletedEndDeviceEventsEventMessage" />
```

```
<wsdl:output name="DeletedEndDeviceEventsResponse" message="tns:ResponseMessage" />
<wsdl:fault name="DeletedEndDeviceEventsFault" message="tns:FaultMessage" />
  </wsdl:operation>
  </wsdl:portType>
<wsdl:binding name="EndDeviceEvents_Binding" type="tns:EndDeviceEvents_Port">
<soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="CreatedEndDeviceEvents">
<soap:operation
   soapAction="http://iec.ch/TC57/2010/EndDeviceEvents/CreatedEndDeviceEvents"
   style="document" />
<wsdl:input name="CreatedEndDeviceEventsRequest">
<soap:body use="literal" />
  </wsdl:input>
<wsdl:output name="CreatedEndDeviceEventsResponse">
<soap:body use="literal" />
  </wsdl:output>
<wsdl:fault name="CreatedEndDeviceEventsFault">
<soap:fault name="CreatedEndDeviceEventsFault" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
<wsdl:operation name="ChangedEndDeviceEvents">
<soap:operation
   soapAction="http://iec.ch/TC57/2010/EndDeviceEvents/ChangedEndDeviceEvents"
   style="document" />
<wsdl:input name="ChangedEndDeviceEventsRequest">
<soap:body use="literal" />
  </wsdl:input>
<wsdl:output name="ChangedEndDeviceEventsResponse">
<soap:body use="literal" />
  </wsdl:output>
<wsdl:fault name="ChangedEndDeviceEventsFault">
<soap:fault name="ChangedEndDeviceEventsFault" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
<wsdl:operation name="CanceledEndDeviceEvents">
<soap:operation
   soapAction="http://iec.ch/TC57/2010/EndDeviceEvents/CanceledEndDeviceEvents"
   style="document" />
<wsdl:input name="CanceledEndDeviceEventsRequest">
<soap:body use="literal" />
  </wsdl:input>
<wsdl:output name="CanceledEndDeviceEventsResponse">
<soap:body use="literal" />
  </wsdl:output>
<wsdl:fault name="CanceledEndDeviceEventsFault">
<soap:fault name="CanceledEndDeviceEventsFault" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
```

```
<wsdl:operation name="ClosedEndDeviceEvents">
 <soap:operation soapAction="http://iec.ch/TC57/2010/EndDeviceEvents/ClosedEndDeviceEvents"
    style="document" />
 <wsdl:input name="ClosedEndDeviceEventsRequest">
 <soap:body use="literal" />
    </wsdl:input>
 <wsdl:output name="ClosedEndDeviceEventsResponse">
 <soap:body use="literal" />
    </wsdl:output>
 <wsdl:fault name="ClosedEndDeviceEventsFault">
 <soap:fault name="ClosedEndDeviceEventsFault" use="literal" />
    </wsdl:fault>
    </wsdl:operation>
 <wsdl:operation name="DeletedEndDeviceEvents">
 <soap:operation
    soapAction="http://iec.ch/TC57/2010/EndDeviceEvents/DeletedEndDeviceEvents"
    style="document" />
 <wsdl:input name="DeletedEndDeviceEventsRequest">
 <soap:body use="literal" />
    </wsdl:input>
 <wsdl:output name="DeletedEndDeviceEventsResponse">
 <soap:body use="literal" />
    </wsdl:output>
 <wsdl:fault name="DeletedEndDeviceEventsFault">
 <soap:fault name="DeletedEndDeviceEventsFault" use="literal" />
    </wsdl:fault>
    </wsdl:operation>
    </wsdl:binding>
 <wsdl:service name="SendEndDeviceEvents">
 <wsdl:port name="EndDeviceEvents_Port" binding="tns:EndDeviceEvents_Binding">
 <soap:address location="http://iec.ch/TC57/2010/SendEndDeviceEvents" />
    </wsdl:port>
    </wsdl:service>
    </wsdl:definitions>
```

(Receive)EndDeviceEvent

```
  <?xml version="1.0" encoding="UTF-8"?>
 <wsdl:definitions
          name="ReceiveEndDeviceEvents"
         targetNamespace="http://iec.ch/TC57/2011/EndDeviceEvents"
         xmlns:tns="http://iec.ch/TC57/2011/EndDeviceEvents"
         xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
         xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
         xmlns:xs="http://www.w3.org/2001/XMLSchema"
         xmlns:wsi="http://ws-i.org/schemas/conformanceClaim/"
         xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
         xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
         xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
```

```
      xmlns:infoMessage="http://iec.ch/TC57/2011/EndDeviceEventsMessage">

  <wsdl:types>

    <xs:schema targetNamespace="http://iec.ch/TC57/2011/EndDeviceEventsMessage"
               elementFormDefault="qualified">

                <xs:include schemaLocation="xsd/EndDeviceEventsMessage.xsd"/>

    </xs:schema>

  </wsdl:types>

  <!-- Message Definitions -->

  <wsdl:message name="CreatedEndDeviceEventsEventMessage">
    <wsdl:part name="CreatedEndDeviceEventsEventMessage"
element="infoMessage:CreatedEndDeviceEvents"/>
  </wsdl:message>

  <wsdl:message name="ChangedEndDeviceEventsEventMessage">
    <wsdl:part name="ChangedEndDeviceEventsEventMessage"
element="infoMessage:ChangedEndDeviceEvents"/>
  </wsdl:message>

  <wsdl:message name="ClosedEndDeviceEventsEventMessage">
    <wsdl:part name="ClosedEndDeviceEventsEventMessage"
element="infoMessage:ClosedEndDeviceEvents"/>
  </wsdl:message>

  <wsdl:message name="CanceledEndDeviceEventsEventMessage">
    <wsdl:part name="CanceledEndDeviceEventsEventMessage"
element="infoMessage:CanceledEndDeviceEvents"/>
  </wsdl:message>

  <wsdl:message name="DeletedEndDeviceEventsEventMessage">
    <wsdl:part name="DeletedEndDeviceEventsEventMessage"
element="infoMessage:DeletedEndDeviceEvents"/>
  </wsdl:message>

  <wsdl:message name="ResponseMessage">
    <wsdl:part name="ResponseMessage"
element="infoMessage:EndDeviceEventsResponseMessage"/>
  </wsdl:message>

  <wsdl:message name="FaultMessage">
    <wsdl:part name="FaultMessage" element="infoMessage:EndDeviceEventsFaultMessage"/>
  </wsdl:message>
```

```
  <!-- Port Definitions -->
  <wsdl:portType name="EndDeviceEvents_Port">

    <wsdl:operation name="CreatedEndDeviceEvents">
      <wsdl:input name="CreatedEndDeviceEventsEvent"
message="tns:CreatedEndDeviceEventsEventMessage"/>
      <wsdl:output name="CreatedEndDeviceEventsResponse"
message="tns:ResponseMessage"/>
      <wsdl:fault name="CreatedEndDeviceEventsFault" message="tns:FaultMessage"/>
    </wsdl:operation>

    <wsdl:operation name="ChangedEndDeviceEvents">
      <wsdl:input name="ChangedEndDeviceEventsEvent"
message="tns:ChangedEndDeviceEventsEventMessage"/>
      <wsdl:output name="ChangedEndDeviceEventsResponse"
message="tns:ResponseMessage"/>
      <wsdl:fault name="ChangedEndDeviceEventsFault" message="tns:FaultMessage"/>
    </wsdl:operation>

    <wsdl:operation name="CanceledEndDeviceEvents">
      <wsdl:input name="CanceledEndDeviceEventsEvent"
message="tns:CanceledEndDeviceEventsEventMessage"/>
      <wsdl:output name="CanceledEndDeviceEventsResponse"
message="tns:ResponseMessage"/>
      <wsdl:fault name="CanceledEndDeviceEventsFault" message="tns:FaultMessage"/>
    </wsdl:operation>

    <wsdl:operation name="ClosedEndDeviceEvents">
      <wsdl:input name="ClosedEndDeviceEventsEvent"
message="tns:ClosedEndDeviceEventsEventMessage"/>
      <wsdl:output name="ClosedEndDeviceEventsResponse"
message="tns:ResponseMessage"/>
      <wsdl:fault name="ClosedEndDeviceEventsFault" message="tns:FaultMessage"/>
    </wsdl:operation>

    <wsdl:operation name="DeletedEndDeviceEvents">
      <wsdl:input name="DeletedEndDeviceEventsEvent"
message="tns:DeletedEndDeviceEventsEventMessage"/>
      <wsdl:output name="DeletedEndDeviceEventsResponse"
message="tns:ResponseMessage"/>
      <wsdl:fault name="DeletedEndDeviceEventsFault" message="tns:FaultMessage"/>
    </wsdl:operation>

  </wsdl:portType>

  <wsdl:binding name="EndDeviceEvents_Binding" type="tns:EndDeviceEvents_Port">

    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
```

```
    <wsdl:operation name="CreatedEndDeviceEvents">
      <soap:operation
soapAction="http://iec.ch/TC57/2011/EndDeviceEvents/CreatedEndDeviceEvents"
style="document"/>
      <wsdl:input name="CreatedEndDeviceEventsEvent">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="CreatedEndDeviceEventsResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="CreatedEndDeviceEventsFault">
        <soap:fault name="CreatedEndDeviceEventsFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="ChangedEndDeviceEvents">
      <soap:operation
soapAction="http://iec.ch/TC57/2011/EndDeviceEvents/ChangedEndDeviceEvents"
style="document"/>
      <wsdl:input name="ChangedEndDeviceEventsEvent">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="ChangedEndDeviceEventsResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ChangedEndDeviceEventsFault">
        <soap:fault name="ChangedEndDeviceEventsFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="CanceledEndDeviceEvents">
      <soap:operation
soapAction="http://iec.ch/TC57/2011/EndDeviceEvents/CanceledEndDeviceEvents"
style="document"/>
      <wsdl:input name="CanceledEndDeviceEventsEvent">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="CanceledEndDeviceEventsResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="CanceledEndDeviceEventsFault">
        <soap:fault name="CanceledEndDeviceEventsFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="ClosedEndDeviceEvents">
      <soap:operation
soapAction="http://iec.ch/TC57/2011/EndDeviceEvents/ClosedEndDeviceEvents"
style="document"/>
      <wsdl:input name="ClosedEndDeviceEventsEvent">
```

```
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="ClosedEndDeviceEventsResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ClosedEndDeviceEventsFault">
        <soap:fault name="ClosedEndDeviceEventsFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="DeletedEndDeviceEvents">
      <soap:operation
  soapAction="http://iec.ch/TC57/2011/EndDeviceEvents/DeletedEndDeviceEvents"
  style="document"/>
      <wsdl:input name="DeletedEndDeviceEventsEvent">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="DeletedEndDeviceEventsResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="DeletedEndDeviceEventsFault">
        <soap:fault name="DeletedEndDeviceEventsFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="ReceiveEndDeviceEvents">
    <wsdl:port name="EndDeviceEvents_Port" binding="tns:EndDeviceEvents_Binding">
      <soap:address location="http://iec.ch/TC57/2011/ReceiveEndDeviceEvents"/>
    </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```

XSDs

Message XSD

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns="http://www.iec.ch/TC57/2010/schema/message"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.iec.ch/TC57/2010/schema/message"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0.0">
<xs:complexType name="RequestType">
   <xs:annotation>
      <xs:documentation>Request type definition</xs:documentation>
   </xs:annotation>
<xs:sequence>
   <xs:annotation>
       <xs:documentation>Request package is typically used to supply parameters for
'get' requests</xs:documentation>
```

```
      </xs:annotation>
   <xs:element name="StartTime" type="xs:dateTime" minOccurs="0">
      <xs:annotation>
         <xs:documentation>Start time of interest</xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:element name="EndTime" type="xs:dateTime" minOccurs="0">
      <xs:annotation>
         <xs:documentation>End time of interest</xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:element name="Option" type="OptionType" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
         <xs:documentation>Request type specialization</xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:element name="ID" type="xs:string" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
         <xs:documentation>Object ID for request</xs:documentation>
      </xs:annotation>
   </xs:element>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded">
         <xs:annotation>
            <xs:documentation>This can be a CIM profile defined as an XSD with a CIM-
specific namespace</xs:documentation>
         </xs:annotation>
      </xs:any>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="ReplyType">
   <xs:annotation>
      <xs:documentation>Reply type definition</xs:documentation>
   </xs:annotation>
   <xs:sequence>
      <xs:annotation>
         <xs:documentation>Reply package is used to confirm success or report
errors</xs:documentation>
      </xs:annotation>
   <xs:element name="Result">
      <xs:annotation>
         <xs:documentation>Reply code: OK, PARTIAL or FAILED</xs:documentation>
      </xs:annotation>
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="OK"/>
            <xs:enumeration value="PARTIAL"/>
            <xs:enumeration value="FAILED"/>
```

```
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
    <xs:element name="Error" type="ErrorType" minOccurs="0" maxOccurs="unbounded">
        <xs:annotation>
            <xs:documentation>Reply details describing one or more
errors</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="ID" type="xs:string" minOccurs="0" maxOccurs="unbounded">
        <xs:annotation>
            <xs:documentation>Resulting transaction ID (usually consequence of
create)</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="operationId" type="xs:integer" minOccurs="0">
        <xs:annotation>
            <xs:documentation>The reply.operationId provides the unique identifier of
the Operation for which this reply.result is relevant. Thus, it is assumed that this
is a partial reply in direct response to one of the operations contained in an
OperationSet request.</xs:documentation>
        </xs:annotation>
    </xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="PayloadType">
    <xs:annotation>
        <xs:documentation>Payload container</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:choice>
            <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>For XML payloads, usually CIM profiles defined using
an XSD in a profile-specific namespace.</xs:documentation>
                </xs:annotation>
            </xs:any>
            <xs:element name="OperationSet" type="OperationSet" minOccurs="0"/>
            <xs:element name="Compressed" type="xs:string" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>For compressed and/or binary, uuencoded
payloads</xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:choice>
        <xs:element name="Format" type="xs:string" minOccurs="0">
            <xs:annotation>
```

```
            <xs:documentation>Hint as to format of payload, e.g. XML, RDF, SVF,
BINARY, PDF, ...</xs:documentation>
         </xs:annotation>
      </xs:element>
   </xs:sequence>
</xs:complexType>
<xs:simpleType name="NounType">
   <xs:restriction base="xs:string">
      <!-- insert nouns from parts 1-8 here -->
      <!-- nouns from "61958-9_MeterReadingAndControl_2ed-working-draft-
20110412.docx" -->
         <xs:enumeration value="AuxiliaryAgreementConfig"/>
         <xs:enumeration value="ComModuleConfig"/>
         <xs:enumeration value="CustomerAccountConfig"/>
         <xs:enumeration value="CustomerAgreementConfig"/>
         <xs:enumeration value="CustomerConfig"/>
         <xs:enumeration value="CustomerMeterDataSet"/>
         <xs:enumeration value="EndDeviceConfig"/>
         <xs:enumeration value="EndDeviceControls"/>
         <xs:enumeration value="EndDeviceEvents"/>
         <xs:enumeration value="EndDeviceFirmware"/>
         <xs:enumeration value="EndDeviceGroups"/>
         <xs:enumeration value="MasterDataLinkageConfig"/>
         <xs:enumeration value="MeterConfig"/>
         <xs:enumeration value="MeterReadings"/>
         <xs:enumeration value="MeterReadSchedule"/>
         <xs:enumeration value="MeterServiceRequest"/>
         <xs:enumeration value="PricingStructureConfig"/>
         <xs:enumeration value="ReceiptRecord"/>
         <xs:enumeration value="ServiceCategoryConfig"/>
         <xs:enumeration value="ServiceLocationConfig"/>
         <xs:enumeration value="ServiceSupplierConfig"/>
         <xs:enumeration value="TransactionRecord"/>
         <xs:enumeration value="UsagePointConfig"/>
         <xs:enumeration value="UsagePointGroups"/>
         <xs:enumeration value="UsagePointLocationConfig"/>
      <!-- insert nouns parts 10-12 here" -->
      </xs:restriction>
   </xs:simpleType>
   <xs:complexType name="OperationSet">
      <xs:annotation>
         <xs:documentation>Each operation set is a collection of operations that may
require operational-integrity and/or sequence control.</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="enforceMsgSequence" type="xs:boolean" minOccurs="0">
            <xs:annotation>
```

```
                <xs:documentation>If set to TRUE, the Operation.##other messages must
be processed in the sequence presented. If omitted, assume FALSE.</xs:documentation>
             </xs:annotation>
          </xs:element>
          <xs:element name="enforceTransactionalIntegrity" type="xs:boolean"
minOccurs="0">
             <xs:annotation>
                <xs:documentation>Set to TRUE when all of the Operation.##other
messages must be processed successfully or else the entire message set must be rolled
back. If omitted, assume TRUE.</xs:documentation>
             </xs:annotation>
          </xs:element>
          <xs:element name="Operation" minOccurs="0" maxOccurs="unbounded">
             <xs:annotation>
                 <xs:documentation>For master data set synchronization XML
payloads.</xs:documentation>
             </xs:annotation>
             <xs:complexType>
                 <xs:sequence>
                    <xs:element name="operationId" type="xs:integer">
                        <xs:annotation>
                           <xs:documentation>The payload.operation.operationId
provides the unique identifier (within the OperationSet) of the Operation for the
purpose of reference in subsequent messages (e.g. OperationSet
reply).</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="noun" minOccurs="0"
type="xs:normalizedString">
                    <!-- <xs:element name="noun" minOccurs="0" type="NounType"> -->
                       <xs:annotation>
                           <xs:documentation>The payload.operation.##other also
identifies the noun, this element is optionally supplied to simplify
processing.</xs:documentation>
                    </xs:annotation>
                 </xs:element>
                 <xs:element name="verb" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>"create", "delete", "change",
etc.</xs:documentation>
                    </xs:annotation>
                 <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="cancel"/>
                        <xs:enumeration value="canceled"/>
                        <xs:enumeration value="change"/>
                        <xs:enumeration value="changed"/>
                        <xs:enumeration value="create"/>
                        <xs:enumeration value="created"/>
```

```
                        <xs:enumeration value="close"/>
                        <xs:enumeration value="closed"/>
                        <xs:enumeration value="delete"/>
                        <xs:enumeration value="deleted"/>
                        <xs:enumeration value="get"/>
                        <xs:enumeration value="show"/>
                        <xs:enumeration value="reply"/>
                        <xs:enumeration value="subscribe"/>
                        <xs:enumeration value="unsubscribe"/>
                        <xs:enumeration value="execute"/>
                        <xs:enumeration value="report"/>
                        <xs:enumeration value="stop"/>
                        <xs:enumeration value="terminate"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:element>
               <xs:element name="elementOperation" type="xs:boolean" default="false"
minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>TRUE if the verb is operating at the element
level. In such a case, the verb is to be applied to the elements populated in the
payload.operation.##other below. If omitted, assume FALSE.</xs:documentation>
                  </xs:annotation>
               </xs:element>
               <xs:any namespace="##other" processContents="skip" minOccurs="0">
                  <xs:annotation>
                     <xs:documentation>An XML payload which carries a CIM profile
defined using an XSD in a profile-specific namespate. Individual payloads are used
collectively to create a series of related operations. See the "enforce" boolean flags
in the header for instructions on how to process these messages.</xs:documentation>
                  </xs:annotation>
               </xs:any>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="ReplayDetectionType">
   <xs:annotation>
      <xs:documentation>Used to detect and prevent replay attacks</xs:documentation>
   </xs:annotation>
   <xs:sequence>
         <xs:element name="Nonce" type="xs:string"/>
         <xs:element name="Created" type="xs:dateTime"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="UserType">
      <xs:annotation>
```

```
         <xs:documentation>User type definition</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="UserID" type="xs:string">
            <xs:annotation>
               <xs:documentation>User identifier</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="Organization" type="xs:string">
            <xs:annotation>
               <xs:documentation>User parent organization
identifier</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="HeaderType">
      <xs:annotation>
         <xs:documentation>Message header type definition</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:annotation>
            <xs:documentation>Message header contains control and descriptive
information about the message.</xs:documentation>
         </xs:annotation>
         <xs:element name="Verb">
            <xs:annotation>
               <xs:documentation>This enumerated list of verbs that can be used to
form message types in compliance with the IEC 61968 standard.</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
               <xs:restriction base="xs:string">
                  <xs:enumeration value="cancel"/>
                  <xs:enumeration value="canceled"/>
                  <xs:enumeration value="change"/>
                  <xs:enumeration value="changed"/>
                  <xs:enumeration value="create"/>
                  <xs:enumeration value="created"/>
                  <xs:enumeration value="close"/>
                  <xs:enumeration value="closed"/>
                  <xs:enumeration value="delete"/>
                  <xs:enumeration value="deleted"/>
                  <xs:enumeration value="get"/>
                  <xs:enumeration value="show"/>
                  <xs:enumeration value="reply"/>
                  <xs:enumeration value="subscribe"/>
                  <xs:enumeration value="unsubscribe"/>
                  <xs:enumeration value="execute"/>
                  <xs:enumeration value="report"/>
```

```
                    <xs:enumeration value="stop"/>
                    <xs:enumeration value="terminate"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="Noun" type="xs:normalizedString">
            <!-- <xs:element name="Noun" type="NounType"> -->
            <xs:annotation>
                <xs:documentation>The Noun of the Control Area identifies the main
subject of the message type, typically a real world object defined in the
CIM.</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="Revision" type="xs:string" minOccurs="0">
            <xs:annotation>
                <xs:documentation>Revision level of the message
type.</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="ReplayDetection" type="ReplayDetectionType" minOccurs="0">
            <xs:annotation>
                <xs:documentation>Use to introduce randomness in the message to
enhance effectiveness of encryption</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="Context" minOccurs="0">
            <xs:annotation>
                <xs:documentation>Intended context for information usage, e.g.
PRODUCTION, TESTING, TRAINING, ...</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="PRODUCTION"/>
                    <xs:enumeration value="TESTING"/>
                    <xs:enumeration value="DEVELOPMENT"/>
                    <xs:enumeration value="STUDY"/>
                    <xs:enumeration value="TRAINING"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="Timestamp" type="xs:dateTime" minOccurs="0">
            <xs:annotation>
                <xs:documentation>Application level relevant time and date for when
this instance of the message type was produced. This is not intended to be used by
middleware for message management.</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="Source" type="xs:string" minOccurs="0">
```

```
            <xs:annotation>
               <xs:documentation>Source system or application that sends the
message</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="AsyncReplyFlag" type="xs:boolean" minOccurs="0">
            <xs:annotation>
               <xs:documentation>Indicates whether or not reply should be
asynchronous</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="ReplyAddress" type="xs:string" minOccurs="0">
            <xs:annotation>
               <xs:documentation>Address to be used for asynchronous replies,
typically a URL/topic/queue.</xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="AckRequired" type="xs:boolean" minOccurs="0">
            <xs:annotation>
               <xs:documentation>Indicates whether or not an acknowledgement is
required</xs:documentation>
            </xs:annotation>
         </xs:element>
            <xs:element name="User" type="UserType" minOccurs="0">
               <xs:annotation>
                  <xs:documentation>User information of the
sender</xs:documentation>
               </xs:annotation>
            </xs:element>
            <xs:element name="MessageID" type="xs:string" minOccurs="0">
               <xs:annotation>
                  <xs:documentation>Unique message ID to be used for tracking
messages</xs:documentation>
               </xs:annotation>
            </xs:element>
            <xs:element name="CorrelationID" type="xs:string" minOccurs="0">
               <xs:annotation>
                  <xs:documentation>ID to be used by applications for correlating
replies</xs:documentation>
               </xs:annotation>
            </xs:element>
            <xs:element name="Comment" type="xs:string" minOccurs="0">
               <xs:annotation>
                  <xs:documentation>Optional comment</xs:documentation
               </xs:annotation>
            </xs:element>
            <xs:element name="Property" type="MessageProperty" minOccurs="0"
maxOccurs="unbounded">
               <xs:annotation>
```

```
                    <xs:documentation>Message properties can be used to identify
information needed for extended routing and filtering capabilities</xs:documentation>
                </xs:annotation>
             </xs:element>
             <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
       </xs:complexType>
       <xs:element name="Message" type="MessageType">
          <xs:annotation>
             <xs:documentation>Common IEC 61968 Message Definition</xs:documentation>
          </xs:annotation>
       </xs:element>
       <xs:complexType name="MessageProperty">
          <xs:annotation>
             <xs:documentation>Message properties can be used for extended routing and
filtering</xs:documentation>
          </xs:annotation>
          <xs:sequence>
             <xs:element name="Name" type="xs:string"/>
             <xs:element name="Value" type="xs:string" minOccurs="0"/>
          </xs:sequence>
       </xs:complexType>
   <xs:element name="RequestMessage" type="RequestMessageType">
       <xs:annotation>
          <xs:documentation>Request message structure</xs:documentation>
       </xs:annotation>
   </xs:element>
   <xs:element name="ResponseMessage" type="ResponseMessageType">
       <xs:annotation>
          <xs:documentation>Response message structure</xs:documentation>
       </xs:annotation>
   </xs:element>
   <xs:element name="EventMessage" type="EventMessageType">
       <xs:annotation>
          <xs:documentation>Event message structure</xs:documentation>
       </xs:annotation>
   </xs:element>
<xs:complexType name="MessageType">
   <xs:annotation>
       <xs:documentation>Generic Message Type</xs:documentation>
   </xs:annotation>
   <xs:sequence>
       <xs:element name="Header" type="HeaderType"/>
       <xs:element name="Request" type="RequestType" minOccurs="0"/>
       <xs:element name="Reply" type="ReplyType" minOccurs="0"/>
       <xs:element name="Payload" type="PayloadType" minOccurs="0"/>
   </xs:sequence>
```

```
</xs:complexType>
<xs:complexType name="RequestMessageType">
   <xs:annotation>
       <xs:documentation>Request Message Type, which will typically result in a
ResponseMessage to be returned. This isn typically used to initiate a transaction or a
query request.</xs:documentation>
   </xs:annotation>
   <xs:sequence>
       <xs:element name="Header" type="HeaderType"/>
       <xs:element name="Request" type="RequestType" minOccurs="0"/>
       <xs:element name="Payload" type="PayloadType" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="ResponseMessageType">
   <xs:annotation>
       <xs:documentation>Response MessageType, typically used to reply to a
RequestMessage</xs:documentation>
   </xs:annotation>
   <xs:sequence>
       <xs:element name="Header" type="HeaderType"/>
       <xs:element name="Reply" type="ReplyType"/>
       <xs:element name="Payload" type="PayloadType" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="FaultMessageType">
   <xs:annotation>
       <xs:documentation>Fault Messsage Type, which is used in cases where the
incoming message (including the header) can not be parsed</xs:documentation>
   </xs:annotation>
</xs:complexType>
<xs:complexType name="EventMessageType">
   <xs:annotation>
       <xs:documentation>Event Message Type, which is used to indicate a condition of
potential interest.</xs:documentation>
   </xs:annotation>
   <xs:sequence>
       <xs:element name="Header" type="HeaderType"/>
       <xs:element name="Payload" type="PayloadType" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="ErrorType">
   <xs:annotation>
       <xs:documentation>Error Structure</xs:documentation>
   </xs:annotation>
   <xs:sequence>
       <xs:element name="code" type="xs:string">
          <xs:annotation>
              <xs:documentation>Application defined error code</xs:documentation>
          </xs:annotation>
```

```
      </xs:element>
      <xs:element name="level" minOccurs="0">
         <xs:annotation>
            <xs:documentation>Severity level, e.g. INFORMATIVE, WARNING, FATAL,
CATASTROPHIC</xs:documentation>
         </xs:annotation>
         <xs:simpleType>
            <xs:restriction base="xs:string">
               <xs:enumeration value="INFORM"/>
               <xs:enumeration value="WARNING"/>
               <xs:enumeration value="FATAL"/>
               <xs:enumeration value="CATASTROPHIC"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:element>
      <xs:element name="reason" minOccurs="0">
         <xs:annotation>
            <xs:documentation>Description of the error</xs:documentation>
         </xs:annotation>
      </xs:element>
      <xs:element name="details" type="xs:string" minOccurs="0">
         <xs:annotation>
            <xs:documentation>Free form detailed text description of
error</xs:documentation>
         </xs:annotation>
      </xs:element>
      <xs:element name="xpath" type="xs:QName" minOccurs="0">
         <xs:annotation>
            <xs:documentation>XPath expression to identify specific XML
element</xs:documentation>
         </xs:annotation>
      </xs:element>
      <xs:element name="stackTrace" type="xs:string" minOccurs="0"/>
      <xs:element name="Location" type="LocationType" minOccurs="0"/>
      <xs:element name="object" type="IdentifiedObject" minOccurs="0"/>
      <xs:element name="operationId" type="xs:integer" minOccurs="0">
         <xs:annotation>
            <xs:documentation>The reply.operationId provides the unique identifier of
the Operation for which this reply.result.error is relevant. Thus, it is assumed that
this is an error from one of the operations contained in an OperationSet
request.</xs:documentation>
         </xs:annotation>
      </xs:element>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="OptionType">
   <xs:annotation>
      <xs:documentation>Request options</xs:documentation>
```

```
   </xs:annotation>
   <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="value" type="xs:string" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="LocationType">
   <xs:annotation>
      <xs:documentation>Process location where error was
encountered</xs:documentation>
   </xs:annotation>
   <xs:sequence>
      <xs:element name="node" type="xs:string" minOccurs="0">
         <xs:annotation>
            <xs:documentation>Name of the pipeline/branch/route node where error
occurred</xs:documentation>
         </xs:annotation>
      </xs:element>
      <xs:element name="pipeline" type="xs:string" minOccurs="0">
         <xs:annotation>
            <xs:documentation>Name of the pipeline where error occurred (if
applicable)</xs:documentation>
         </xs:annotation>
      </xs:element>
      <xs:element name="stage" type="xs:string" minOccurs="0">
         <xs:annotation>
            <xs:documentation>Name of the stage where error occurred (if
applicable)</xs:documentation>
         </xs:annotation>
      </xs:element>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="IdentifiedObject">
   <xs:annotation>
      <xs:documentation>From CIM</xs:documentation>
   </xs:annotation>
   <xs:sequence>
      <xs:element name="mRID" minOccurs="0"/>
      <xs:element name="Name" type="Name" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="objectType" type="xs:string" minOccurs="0">
         <xs:annotation>
            <xs:documentation>CIM class name that classiifies the identified
object</xs:documentation>
         </xs:annotation>
      </xs:element>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="NameType">
   <xs:annotation>
```

```
        <xs:documentation>From CIM</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="name"/>
        <xs:element name="description" minOccurs="0"/>
        <xs:element name="NameTypeAuthority" type="NameTypeAuthority" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="Name">
    <xs:annotation>
        <xs:documentation>From CIM</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="name"/>
        <xs:element name="NameType" type="NameType" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="NameTypeAuthority">
    <xs:annotation>
        <xs:documentation>From CIM</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="name"/>
        <xs:element name="description" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="FaultMessage" type="FaultMessageType"/>
<xs:element name="adverbTypes">
    <xs:simpleType>
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    </xs:element>
</xs:schema>
```

EndDeviceEventsMessage.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://iec.ch/TC57/2011/EndDeviceEventsMessage"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msg="http://iec.ch/TC57/2011/schema/message"
xmlns:obj="http://iec.ch/TC57/2011/EndDeviceEvents#"
targetNamespace="http://iec.ch/TC57/2011/EndDeviceEventsMessage"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0.0">
       <!-- Base Message Definitions -->
       <xs:import namespace="http://iec.ch/TC57/2011/schema/message"
schemaLocation="Message.xsd"/>
```

```
	<!-- CIM Information Object Definition -->
	<xs:import namespace="http://iec.ch/TC57/2011/EndDeviceEvents#"
schemaLocation="EndDeviceEvents.xsd"/>
	<!-- PayloadType Definition -->
	<xs:complexType name="EndDeviceEventsPayloadType">
		<xs:sequence>
			<xs:element ref="obj:EndDeviceEvents"/>
			<xs:element name="OperationSet" type="msg:OperationSet"
minOccurs="0"/>
			<xs:element name="Compressed" type="xs:string" minOccurs="0">
				<xs:annotation>
					<xs:documentation>For compressed and/or binary,
uuencoded payloads</xs:documentation>
				</xs:annotation>
			</xs:element>
			<xs:element name="Format" type="xs:string" minOccurs="0">
				<xs:annotation>
					<xs:documentation>Hint as to format of payload, e.g.
XML, RDF, SVF, BINARY, PDF, ...</xs:documentation>
				</xs:annotation>
			</xs:element>
		</xs:sequence>
	</xs:complexType>
	<!-- Message Types -->
	<!-- RequestMessageType -->
	<xs:complexType name="EndDeviceEventsRequestMessageType">
		<xs:sequence>
			<xs:element name="Header" type="msg:HeaderType"/>
			<xs:element name="Request" type="msg:RequestType" minOccurs="0"/>
			<xs:element name="Payload" type="tns:EndDeviceEventsPayloadType"
minOccurs="0"/>
		</xs:sequence>
	</xs:complexType>
	<!-- ResponseMessageType -->
	<xs:complexType name="EndDeviceEventsResponseMessageType">
		<xs:sequence>
			<xs:element name="Header" type="msg:HeaderType"/>
			<xs:element name="Reply" type="msg:ReplyType"/>
			<xs:element name="Payload" type="tns:EndDeviceEventsPayloadType"
minOccurs="0"/>
		</xs:sequence>
	</xs:complexType>
	<!-- EventMessageType -->
	<xs:complexType name="EndDeviceEventsEventMessageType">
		<xs:sequence>
			<xs:element name="Header" type="msg:HeaderType"/>
			<xs:element name="Payload" type="tns:EndDeviceEventsPayloadType"
minOccurs="0"/>
		</xs:sequence>
```

```
	</xs:complexType>
	<!-- FaultMessageType -->
	<xs:complexType name="EndDeviceEventsFaultMessageType">
		<xs:sequence>
			<xs:element name="Reply" type="msg:ReplyType"/>
		</xs:sequence>
	</xs:complexType>
	<xs:element name="CreateEndDeviceEvents"
type="tns:EndDeviceEventsRequestMessageType"/>
	<xs:element name="ChangeEndDeviceEvents"
type="tns:EndDeviceEventsRequestMessageType"/>
	<xs:element name="CancelEndDeviceEvents"
type="tns:EndDeviceEventsRequestMessageType"/>
	<xs:element name="CloseEndDeviceEvents"
type="tns:EndDeviceEventsRequestMessageType"/>
	<xs:element name="DeleteEndDeviceEvents"
type="tns:EndDeviceEventsRequestMessageType"/>
	<xs:element name="CreatedEndDeviceEvents"
type="tns:EndDeviceEventsEventMessageType"/>
	<xs:element name="ChangedEndDeviceEvents"
type="tns:EndDeviceEventsEventMessageType"/>
	<xs:element name="CanceledEndDeviceEvents"
type="tns:EndDeviceEventsEventMessageType"/>
	<xs:element name="ClosedEndDeviceEvents"
type="tns:EndDeviceEventsEventMessageType"/>
	<xs:element name="DeletedEndDeviceEvents"
type="tns:EndDeviceEventsEventMessageType"/>
	<xs:element name="EndDeviceEventsResponseMessage"
type="tns:EndDeviceEventsResponseMessageType"/>
	<xs:element name="EndDeviceEventsFaultMessage"
type="tns:EndDeviceEventsFaultMessageType"/>
</xs:schema>
```

The Electric Power Research Institute Inc., (EPRI, www.epri.com) conducts research and development relating to the generation, delivery and use of electricity for the benefit of the public. An independent, nonprofit organization, EPRI brings together its scientists and engineers as well as experts from academia and industry to help address challenges in electricity, including reliability, efficiency, health, safety and the environment. EPRI also provides technology, policy and economic analyses to drive long-range research and development planning, and supports research in emerging technologies. EPRI's members represent more than 90 percent of the electricity generated and delivered in the United States, and international participation extends to 40 countries. EPRI's principal offices and laboratories are located in Palo Alto, Calif.; Charlotte, N.C.; Knoxville, Tenn.; and Lenox, Mass.

Together...Shaping the Future of Electricity

Program:
IntelliGrid

1024445

Electric Power Research Institute
3420 Hillview Avenue, Palo Alto, California 94304-1338 • PO Box 10412, Palo Alto, California 94303-0813 USA
800.313.3774 • 650.855.2121 • askepri@epri.com • www.epri.com