EPRI | ELECTRIC POWER RESEARCH INSTITUTE

Common Information Model Meter Tamper Detection Interoperability Test Procedure

EPRI CIM Interoperability Test Procedure (ECITP) 2.07

2011 TECHNICAL REPORT

Common Information Model Meter Tamper Detection Interoperability Test Procedure

EPRI CIM Interoperability Test Procedure (ECITP) 2.07

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

1024446
Final Report, November 2011

DISCLAIMER OF WARRANTIES AND LIMITATION OF LIABILITIES

THE FOLLOWING ORGANIZATIONS, UNDER CONTRACT TO EPRI, PREPARED THIS REPORT:

EnerNex

Boreas Group

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, Tennessee 37932

Principal Investigators
K. Stefferud
B. Muschlitz

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

Principal Investigators
R. Sarfi
B. Boswell

This report describes research sponsored by EPRI.

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

Common Information Model Meter Tamper Detection Interoperability Test Procedure: EPRI CIM Interoperability Test Procedure (ECITP) 2.07.
EPRI, Palo Alto, CA: 2011.
1024446.

Abstract

The Common Information Model (CIM) Meter Tamper Detection Interoperability Test Procedure is one in a series of EPRI 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 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 and Certification Committee (SGTCC) in its Interoperability Process Reference Manual (IPRM) whose goal is the formalized validation of the interoperability of combinations of vendor products

- Organizations sponsoring routine standards-validation CIM interoperability testing

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

Keywords
Smart meters
Common Information Model (CIM)
Interoperability Test Procedures (Tips)
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 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 1-1This document, CIM Meter Tamper Detection Interoperability Test Procedure, comprises one of component procedures of the 61968-9 collection of Test Procedures.

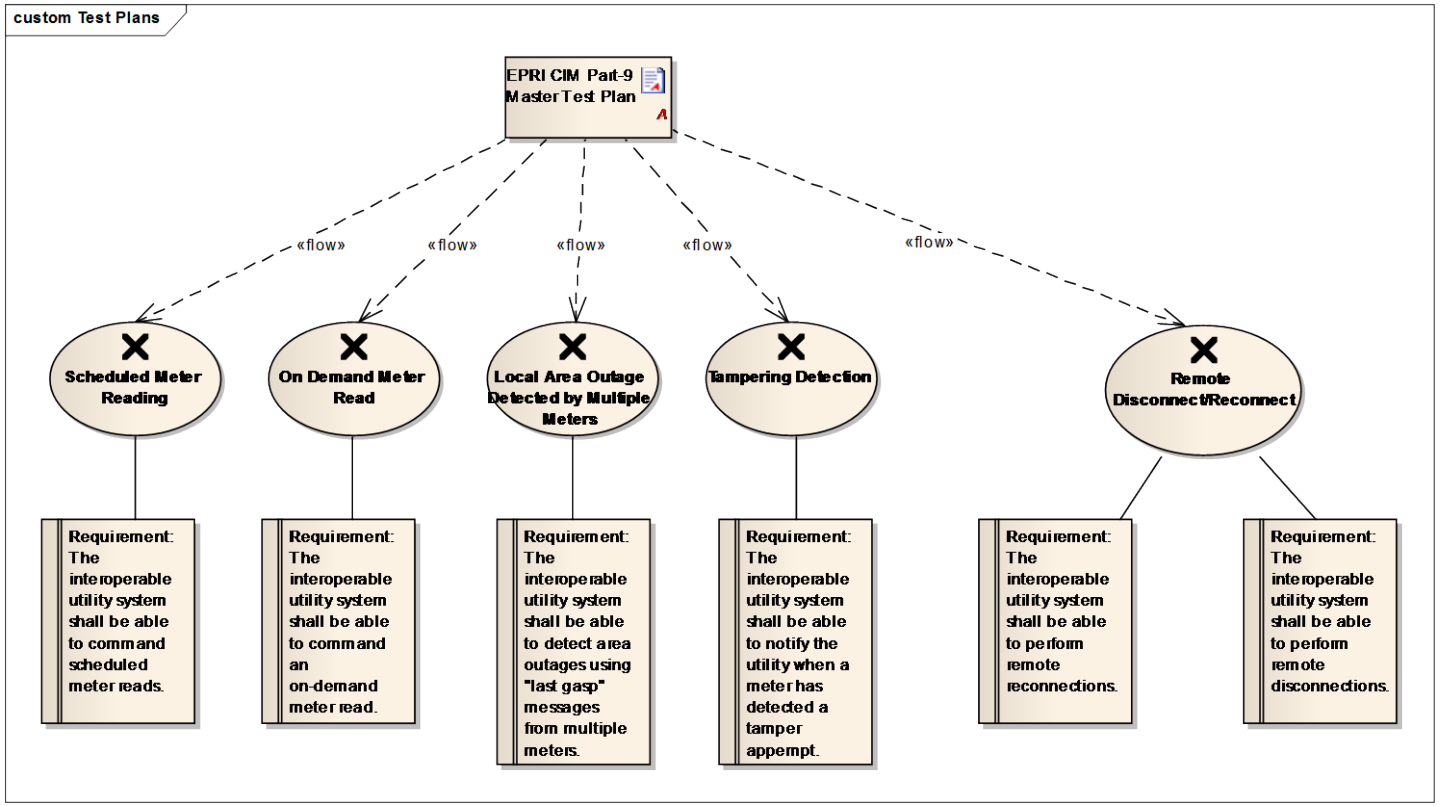

Figure 1-1
The Scope of the EPRI CIM 61968-9 Test Plans

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.

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 cases from smartgridipedia:

http://www.smartgridipedia.org/index.php/B3_-_Detect_Theft

B3 - Scenario 1 - Meter removal

B3 - Scenario 3 - Meter is inverted

B3 - Scenario 4 - Meter bypass detection at the Meter

B3 - Scenario 5 - Physical tamper detection

B3 - Scenario 6 - Unauthorized meter location change

- Use Case B3 AMI-ENT Scenario 1 – Meter Removal REQ-B3001 – Send meter event
- Use Case B3 AMI-ENT Scenario 3 – Meter inverted REQ-B3001 – Send meter event
- Use Case B3 AMI-ENT Scenario 4 – Meter bypass detection at the meter REQ-B3001 – Send meter Event
- Use Case B3 AMI-ENT Scenario 5 – Physical tamper detection REQ-B3001 – Send meter event
- Use Case B3 AMI-ENT Scenario 6 – Unauthorized meter location change REQ-B3001 – Send meter event

Section 3: Test Overview

Meter Tamper Detection Nominal Interoperability Test

The ability to detect meter tampering is a basic interoperability requirement of utility systems. Systems tested include the Customer Information System (CIS), or similar system, Meter Data Management System (MDMS), and the Metering System (Head-End). The Meter Tampering Interoperability Test verifies that the MS returns the correct response CreatedEndDeviceEvents message corresponding to the tampering events. The 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. Actual meters, if available, may be used in the test system.

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)
Theft Detection	NO – FLT (Network Operations – Fault Management)

The Meter Tamper Detection Interoperability Test Procedure is one of the set of EPRI CIM demonstration tests. The overall set of demonstration tests is shown in the Figure 1-1 Units Under Test (UUT)

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

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

The goal of the EPRI demonstration tests is to show system-to-system interoperability using IEC standard 61968 messages.

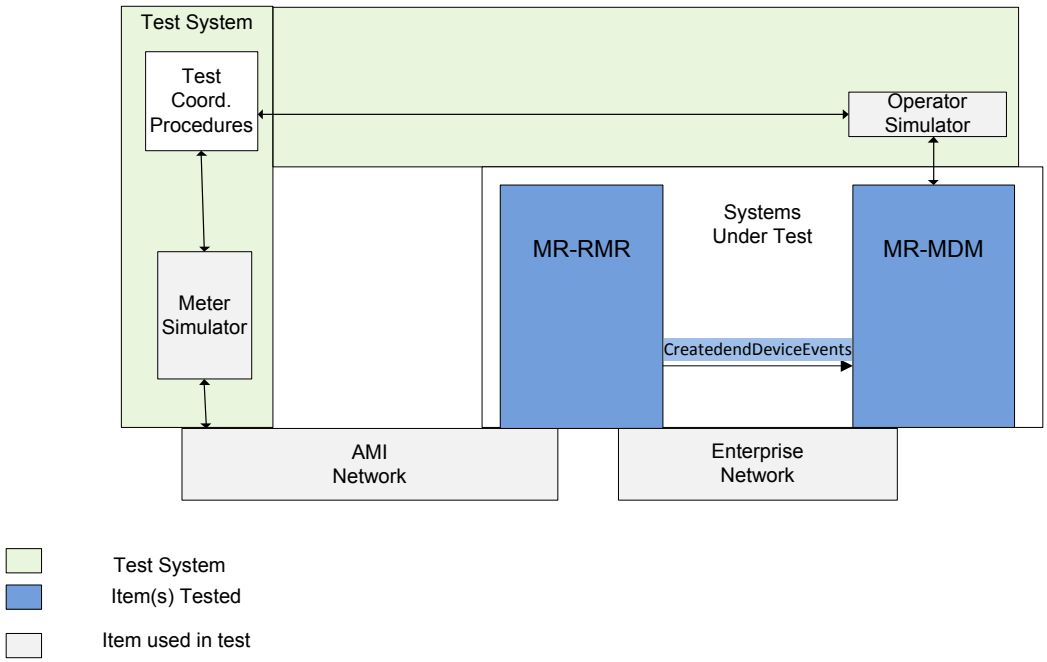

Figure 3-1
Units under Test (UUT) for the Meter Tamper Nominal Interoperability Test

Interface Description

The system interfaces are shown in Figure 3-2 ECITP 2.07 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 3.12.0.257 for the meter tamper detection 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.

Please see Section 6: Test Data for the sample XML format and content of the meter tampering detection event message used for this test.

As shown in Figure 3-2 ECITP 2.07 below, the AMI Head End receives the meter tampering detection event in proprietary non-standard format. The Meter Reading-Meter System as defined in IEC 61968 and as the AMI Head End and MDMS systems in actual implementations, perform event processing and send a tampering detection event message to the CIS.

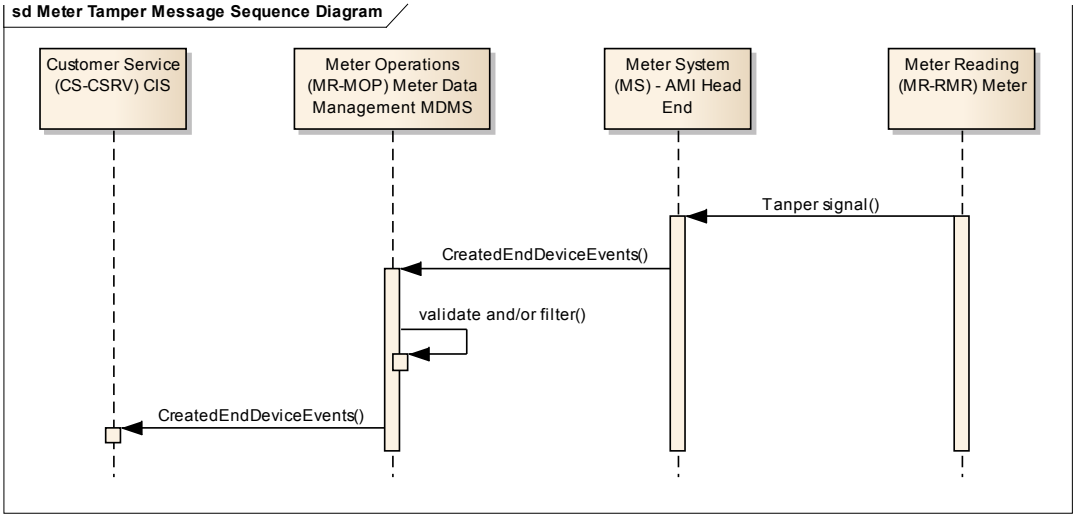

Figure 3-2
ECITP 2.07 Meter Tamper Message Sequence Flow Diagram

Test Description

The ability of the MS (AMI Head End) to detect and report tampering 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 MDMS are used to report a tamper for first an individual and then multiple meters. The systems' response to the simulated tampers 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 3-2 in Universal Modeling Language (UML) format.

The AMI Head-end is stimulated with signals indicating meter tampering. This test verifies that the tampering message is correctly transferred from the head-end to the MDMS and that the Customer Information System observes the tamping event.

CIM standard Part-9 EndDeviceEvent messages carry the tamper indication indications in the <category> element.

Unit or Systems under Test (SUTs/UUTs) are the Metering System (MS), and the Meter Data Management System (MDMS) and the Customer Information System (CIS). The test steps are as follows:

1. The Head-end is stimulated to cause a meter tampering indication to be noted.
2. The Head-end transmit the EndDeviceEvent containing the tamper detection to the MDMD

3. The MDMS Receives and recognizes the tamper detecting
4. The CIS observes the tampering event

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 tamper 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 role, detailed test procedure steps specific to that role are detailed.

MDM Test Steps

Table 4-1
Test Steps for MDM

Step #	Ref. #	Source Component	Destination Component	Detailed Step	Pass/Fail
1	IEC 61968 version 10v30, EndDeviceEvent XSD	Meter or meter simulator	Head-end	Force a tampering event which is sent to head-end	Pass/Fail
2	Use Case B2 AMI ENT REQ0262	Head-end	MDMS	Verify one or more EndDeviceEvent messages are sent with End Device Event code 3.12.0.257. Verify that the MDMS records the Timestamp, Tamper status (event type) and meter ID in the log for each tamper event.	Pass/Fail
3	B3-USE CASE v1.2 050106	MDMS	CIS	Verify one or more EndDeviceEvent messages are sent to the CIS with End Device Event code 3.12.0.257.	Pass/Fail

AMI Head End Test Steps

Table 4-2
Test Steps for AMI

Step #	Ref. #	Source Component	Destination Component	Detailed Step	Pass/Fail
1	IEC 61968 version 10v30, EndDeviceEvent XSD	Meter or meter simulator, via the MDMS	Head-end	Force a tampering event which is sent to head-end	Pass/Fail
2	Use Case B2 AMI ENT REQ0262	Head-end	MDMS	Verify one or more EndDeviceEvent messages is sent with End Device Event code 3.12.0.257. Verify that the MDMS records the Timestamp, Tamper status (event type) and meter ID in the log for each tamper event.	Pass/Fail
3	B3-USE CASE v1.2 050106	MDMS	CIS	Verify one or more EndDeviceEvent messages is sent to the CIS with End Device Event code 3.12.0.257.	Pass/Fail

Section 5: 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 is easy to generate new instances on demand and the service itself is extremely reliable.

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 IEC 61968 Part 100 CD defined message header and EndDeviceEvent payload which are included in Appendix A.

Test data

Meter Tamper sample message which is expected to be produced by the meter and sent to the CIS via a meter head-end and MDMS. Note in some implementations, Meter Tamper EndDeviceEvents messages may be combined with other EndDeviceEvents. Further EndDeviceEvents may be piggybacked on MeterReading messages.

Meter Tampering event

```
<?xml version="1.0" encoding="UTF-8"?>
<CreatedEndDeviceEvents
       xsi:schemaLocation="http://www.iec.ch/TC57/2011/EndDeviceEventsMes
sage EndDeviceEventsMessage.xsd"
       xmlns="http://www.iec.ch/TC57/2011/EndDeviceEventsMessage"
       xmlns:obj="http://iec.ch/TC57/2011/EndDeviceEvents#"
                xmlns:msg="http://www.iec.ch/TC57/2010/schema/message"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
       <Header>
              <msg:Verb>created</msg:Verb>
              <msg:Noun>EndDeviceEvents</msg:Noun>
              <msg:Context>TESTING</msg:Context>
              <msg:Timestamp>2011-04-12T10:00:00Z</msg:Timestamp>
              <msg:Source>HE-001</msg:Source>
              <msg:AsyncReplyFlag>false</msg:AsyncReplyFlag>
              <msg:AckRequired>true</msg:AckRequired>
              <msg:MessageID>ABC-123</msg:MessageID>
              <msg:CorrelationID>Pre-
configured_Request</msg:CorrelationID>
              <msg:Comment>created end device events for testing
tamper</msg:Comment>
       </Header>
       <Payload>
              <obj:EndDeviceEvents>
                     <obj:EndDeviceEvent>
                            <obj:createdDateTime>2011-04-
12T09:30:30Z</obj:createdDateTime>
                            <obj:reason>Electric Meter Alarm Tamper
(General Tamper)</obj:reason>
                            <obj:Assets>
                                   <obj:mRID>3dc53ee5-777e-50b4-8699-
a1c224f45f3d</obj:mRID>
                                   <obj:Names>
                                          <obj:name>Meter23253</obj:name>
                                   </obj:Names>
                            </obj:Assets>
                            <obj:EndDeviceEventType
```

```
ref="3.12.0.257"></obj:EndDeviceEventType>
                    </obj:EndDeviceEvent>
             </obj:EndDeviceEvents>
       </Payload>
</CreatedEndDeviceEvents
```

Test results

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

Appendix A: Detailed Description of Each Interface

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

WSDLs

(Created)EndDeviceEvent

```
<?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/CreatedEndDeviceEve
     nts" 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/ChangedEndDeviceEve
     nts" 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/CanceledEndDeviceEv
     ents" 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/ClosedEndDeviceEven
     ts" 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/DeletedEndDeviceEve
     nts" 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"
     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/CreatedEndDeviceEve
     nts" 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/ChangedEndDeviceEve
     nts" 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/CanceledEndDeviceEv
     ents" 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/ClosedEndDeviceEven
     ts" 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/DeletedEndDeviceEve
     nts" 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="ReceiveEndDeviceEvents">
   <wsdl:port name="EndDeviceEvents_Port"
     binding="tns:EndDeviceEvents_Binding">
        <soap:address
     location="http://iec.ch/TC57/2010/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="http://www.iec.ch/TC57/2010/EndDeviceEventsMessage"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msg="http://www.iec.ch/TC57/2010/schema/message"
xmlns:obj="http://iec.ch/TC57/2010/EndDeviceEvents#"
targetNamespace="http://www.iec.ch/TC57/2010/EndDeviceEventsMessage"
elementFormDefault="qualified" attributeFormDefault="unqualified"
```

```
version="1.0.0">
<!-- Base Message Definitions -->
   <xs:import namespace="http://www.iec.ch/TC57/2010/schema/message"
schemaLocation="Message.xsd"/>
<!-- CIM Information Object Definition -->
   <xs:import namespace="http://iec.ch/TC57/2010/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="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="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="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="EndDeviceEventsRequestMessageType"/>
    <xs:element name="ChangeEndDeviceEvents"
type="EndDeviceEventsRequestMessageType"/>
    <xs:element name="CancelEndDeviceEvents"
type="EndDeviceEventsRequestMessageType"/>
    <xs:element name="CloseEndDeviceEvents"
type="EndDeviceEventsRequestMessageType"/>
    <xs:element name="DeleteEndDeviceEvents"
type="EndDeviceEventsRequestMessageType"/>
    <xs:element name="CreatedEndDeviceEvents"
type="EndDeviceEventsEventMessageType"/>
    <xs:element name="ChangedEndDeviceEvents"
type="EndDeviceEventsEventMessageType"/>
    <xs:element name="CanceledEndDeviceEvents"
type="EndDeviceEventsEventMessageType"/>
    <xs:element name="ClosedEndDeviceEvents"
type="EndDeviceEventsEventMessageType"/>
    <xs:element name="DeletedEndDeviceEvents"
type="EndDeviceEventsEventMessageType"/>
    <xs:element name="EndDeviceEventsResponseMessage"
type="EndDeviceEventsResponseMessageType"/>
    <xs:element name="EndDeviceEventsFaultMessage"
type="EndDeviceEventsFaultMessageType"/>
</xs:schema>
```

Appendix B: Requirements Tested

B3-USE CASE v1.2 050106. Requirement listed fifth: For each tamper event, the meter shall transmit and locally log the following information about the event:

- Timestamp
- Tamper status (event type)
- Meter ID

Use Case B2 AMI ENT REQ0253 - The Meter shall have a unique serial number.

Use Case B2 AMI ENT REQ0254 - The Meter shall be able to detect removal from its socket.

Use Case B2 AMI ENT REQ0255 - The Meter shall differentiate between a meter removal event and a power outage event.

Use Case B2 AMI ENT REQ0259 - The Meter shall send a removal event upon meter removal.

Use Case B2 AMI ENT REQ0260 - The Meter shall create and record tamper events as soon as the event occurs.

Use Case B2 AMI ENT REQ0261 - The AMI Communication Network will treat tampering messages involving safety issues and complete meter failure with a higher priority than normal status messages.

Use Case B2 AMI ENT REQ0263 - The Meter shall record and store a tamper event even when the event has been reported to the AMI Head End.

Use Case B2 AMI ENT REQ0266 - The Meter shall store tamper events for at least 45 days.

Use Case B2 AMI ENT REQ0267 - The Meter shall send unconfirmed tamper event messages to the AMI Head End up meter re-installation.

Use Case B2 AMI ENT REQ0271 - The AMI Enterprise system shall perform energy theft analysis.

Use Case B2 AMI ENT REQ0274 - The Customer Care System shall have access to tamper detection events.

Use Case B2 AMI ENT REQ0275 - The Customer Care System shall contact the Customer when a tamper event has been detected.

Use Case B2 AMI ENT REQ0276 - The AMI Mangement System shall correlate tamper events with work orders and work authorizations.

Off nominal and error requirements

Use Case B2 AMI ENT REQ0262 - The Meter shall retain tamper events in its event log until a confirmation message is received from the AMI Head End.

Use Case B2 AMI ENT REQ0268 - The Meter shall send a re-installation message that is different from messages sent for normal installation or power restoration.

Export Control Restrictions

Access to and use of EPRI Intellectual Property is granted with the specific understanding and requirement that responsibility for ensuring full compliance with all applicable U.S. and foreign export laws and regulations is being undertaken by you and your company. This includes an obligation to ensure that any individual receiving access hereunder who is not a U.S. citizen or permanent U.S. resident is permitted access under applicable U.S. and foreign export laws and regulations. In the event you are uncertain whether you or your company may lawfully obtain access to this EPRI Intellectual Property, you acknowledge that it is your obligation to consult with your company's legal counsel to determine whether this access is lawful. Although EPRI may make available on a case-by-case basis an informal assessment of the applicable U.S. export classification for specific EPRI Intellectual Property, you and your company acknowledge that this assessment is solely for informational purposes and not for reliance purposes. You and your company acknowledge that it is still the obligation of you and your company to make your own assessment of the applicable U.S. export classification and ensure compliance accordingly. You and your company understand and acknowledge your obligations to make a prompt report to EPRI and the appropriate authorities regarding any access to or use of EPRI Intellectual Property hereunder that may be in violation of applicable U.S. or foreign export laws or regulations.

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

1024446

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