Lessons Learned in Implementing IEC 61850 Communications Solutions

Kevin Mahoney and Kevin Coyne
Casco Systems

Brian Waldron
Schweitzer Engineering Laboratories, Inc.

Presented at the
Power and Energy Automation Conference
Spokane, Washington
March 5–6, 2019

Lessons Learned in Implementing IEC 61850 Communications Solutions

Kevin Mahoney and Kevin Coyne, *Casco Systems*
Brian Waldron, *Schweitzer Engineering Laboratories, Inc.*

***Abstract*—One of the main goals of the IEC 61850 standard is to allow for interoperability between products from different manufacturers in a substation. This allows power system operators to decrease their dependence on any one type of intelligent electronic device (IED). However, over the past decade as engineers have implemented IEC 61850 solutions, there has been a variety of communications challenges between IEDs. Common challenges that implementers face are as follows:**

- **Why is the IEC 61850 Manufacturing Message Specification (MMS) connection not active?**
- **Why do report control blocks not activate when the client-and-server connection is active?**
- **Why does some information not update correctly or have an invalid quality?**
- **Why do controls not work when the client-and-server connection is active?**

This paper walks through several examples of how to identify and troubleshoot these problems by using information from Substation Configuration Language (SCL) files, the IEC 61850 data model, and communications captures. This information is analyzed using the open-source packet analytic tool Wireshark. Then, after identifying the problems, the solutions are discussed.

I. Introduction

The IEC 61850 standard describes methods of collecting data and sending control signals from data concentrators and human-machine interfaces to intelligent electronic devices (IEDs). This communication takes place primarily via the Manufacturing Message Specification (MMS) protocol. IEC 61850 offers a feature-rich environment to exchange information between clients and servers. However, the large number of options can make the data transfer more complex. During the tenure and implementation of IEC 61850, integrators have identified many integration challenges. Often, troubleshooting these issues is challenging because many users are not familiar with the intricacies of IEC 61850 that allow a varied feature set for exchanging information.

This paper discusses many of the common challenges of integrating IEC 61850 MMS IEDs, explains how to diagnose problems using packet analysis tools and IEC 61850 browsers, and identifies the settings in the IED or Configured IED Description (CID) file that need to be adjusted to resolve these challenges.

An MMS browser is an engineering tool that determines the data model of an MMS server without an IED capability description (ICD) or CID file. The MMS browser instead uses the self-discovery services in the MMS protocol to identify the IEC 61850 data model. It is also capable of reading and writing to the IED to see real-time values or to change active values.

A packet analysis tool captures Ethernet-based network traffic and decodes it into human-readable fields. This allows for easy analysis of the information that is exchanged between devices. There are paid and open-source options for packet analysis tools and MMS browsers.

Troubleshooting using an IEC 61850 MMS browser almost always makes it easier to see current values and requires less knowledge of the IEC 61850 standard than analyzing network captures does. However, an MMS browser is not always available, and troubleshooting may occur remotely where access to the server is limited. When possible, it is best to have the following information and tools when analyzing MMS communications:

- MMS client configuration
- MMS server configuration
- IEC 61850 MMS browser
- Packet analysis tool

II. Wireshark

The most common network analysis tool is the open-source network capture software, Wireshark. This paper uses example screenshots from this tool for packet analysis.

Ideally, network captures contain all traffic between the client and server. However, that may not always be possible. There are three locations from which to obtain a network capture: the client, the server, and the switch. Provided that all connections in the network are active, captures in each location provide the same information regarding the MMS client-and-server conversation.

In many cases, the location in which a capture is taken does not significantly affect the analytical process discussed in this paper. However, this is not always the case, especially for troubleshooting network communication issues other than MMS. It is best to analyze traffic from a mirrored port on a switch; however, not all switches support port mirroring or they may require a settings change, which is not always easy to do in a commissioning or in-service environment. Captures from clients with Wireshark installed or from a communications capture tool are much easier to obtain.

It is important to use Ethernet packet analysis and MMS browsers as troubleshooting tools and to determine what IEC 61850 data model is active on the server instead of relying on IEC 61850 Substation Configuration Language (SCL) files. Many IEDs do not configure from their CID file but rather from some other settings file. Users are often unfamiliar with how to update IEC 61850 settings on a device or do not realize that the

CID file they think is on an IED is not actually present. This is why using an MMS browser or packet analysis is important for troubleshooting issues when integrating IEC 61850 MMS clients and servers.

III. Connection Problems

Just like any other Ethernet-based protocol that uses a Transmission Control Protocol (TCP) socket connection, in MMS there can be problems establishing a socket and getting the client and server to exchange information. Before examining the finer details of the MMS protocol, it is important to make sure that the client and server can exchange application data.

A. TCP Port Problems

The first thing to confirm if no data are being collected is whether a socket connection has been established. Often, this can be confirmed if the client or server offers communication statistics, like message sent and received counters. If the sent count increments but the received count does not, then likely no socket connection is established. This is often due to an incorrect Internet Protocol (IP) address, a cable unplugged in the network, a firewall preventing communications, or functionality not being enabled on the server.

In many other SCADA protocols, it is common to change the default TCP port, but this configuration parameter is often static with MMS. TCP Port 102 is the default server port, and many configuration tools do not allow this port to be changed. To verify that the correct port is used, take an Ethernet capture. There should be multiple packet transmissions to the server IP address but no response from the server IP address. If the capture looks like Fig. 1 where there are no server responses, or a three-way TCP socket connection, then investigate the previously mentioned issues.

B. Connection-Oriented Transport Protocol (COTP) Connection Issues

A problem that occasionally prevents MMS communications from being exchanged is that the MMS transport protocols do not connect correctly. Most protocols are encapsulated in TCP, part of the IP family (often referred to as TCP/IP) [1], which is in wide use (see Table I). However, when MMS was standardized in 1990, it was built on the Open Systems Interconnection (OSI) [2] protocol suite (see Table II), which was challenging to implement and has since become obsolete [3].

Table I
IP Family Model

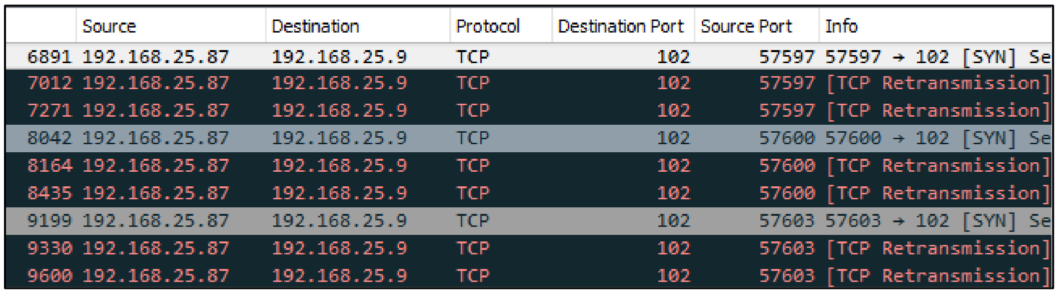
Application	Association Control Service Element—ISO 8649/8650
Presentation	Connection-Oriented Presentation—ISO 8822/8823 Abstract Syntax Notation—ISO 8824/8825
Session	Connection-Oriented Session—ISO 8326/8327
Transport	ISO Transport Over TCP—RFC 1006 TCP—RFC 793
Network	Internet Control Message Protocol—RFC 792 IP—RFC 791 Address Resolution Protocol—RFC 826
Link	IP Datagrams Over Ethernet—RFC 894 Media Access Control—ISO 8802-3 [Ethernet]
Physical	Ethernet

Table II
OSI Model

Application	Association Control Service Element—ISO 8649/8650
Presentation	Connection-Oriented Presentation—ISO 8822/8823 Abstract Syntax Notation—ISO 8824/8825
Session	Connection-Oriented Session—ISO 8326/8327
Transport	Connection-Oriented Transport—ISO 8072/8073
Network	Connectionless Network—ISO 8348
Link	Media Access Control—ISO 8802-3 (Ethernet) Media Access Control—ISO 8802-4 (Token Ring)
Physical	Ethernet Token Ring

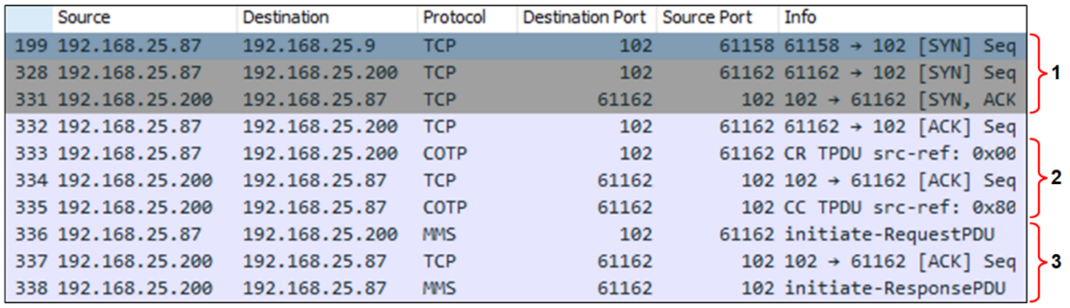
Because the underlying transport protocols became obsolete in modern networking, a new version of MMS was specified that is largely implemented inside of TCP. As a result, there are other protocols that need to connect after a TCP socket is established in order for MMS communications to occur. Sometimes the COTP that carries MMS does not connect successfully. When an MMS connection starts, there are three main steps that occur, as shown in Fig. 2:

1. A TCP socket connection is created.
2. A COTP connection is created.
3. MMS initiates data exchange.



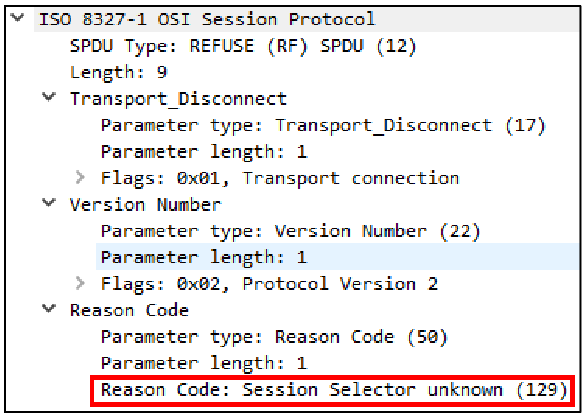
```
      Source         Destination   Protocol  Destination Port  Source Port  Info
6891  192.168.25.87  192.168.25.9  TCP                    102        57597  57597 → 102 [SYN] Se
7012  192.168.25.87  192.168.25.9  TCP                    102        57597  [TCP Retransmission]
7271  192.168.25.87  192.168.25.9  TCP                    102        57597  [TCP Retransmission]
8042  192.168.25.87  192.168.25.9  TCP                    102        57600  57600 → 102 [SYN] Se
8164  192.168.25.87  192.168.25.9  TCP                    102        57600  [TCP Retransmission]
8435  192.168.25.87  192.168.25.9  TCP                    102        57600  [TCP Retransmission]
9199  192.168.25.87  192.168.25.9  TCP                    102        57603  57603 → 102 [SYN] Se
9330  192.168.25.87  192.168.25.9  TCP                    102        57603  [TCP Retransmission]
9600  192.168.25.87  192.168.25.9  TCP                    102        57603  [TCP Retransmission]
```

Fig. 1. TCP socket connection attempts



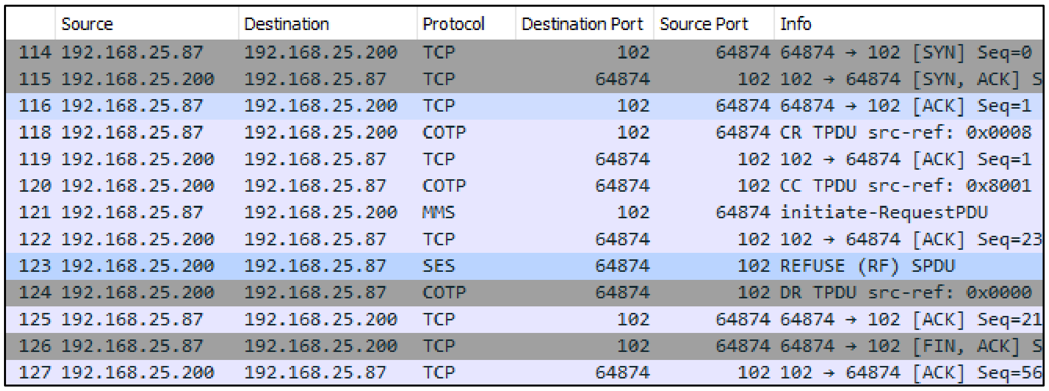
	Source	Destination	Protocol	Destination Port	Source Port	Info
199	192.168.25.87	192.168.25.9	TCP	102	61158	61158 → 102 [SYN] Seq
328	192.168.25.87	192.168.25.200	TCP	102	61162	61162 → 102 [SYN] Seq
331	192.168.25.200	192.168.25.87	TCP	61162	102	102 → 61162 [SYN, ACK
332	192.168.25.87	192.168.25.200	TCP	102	61162	61162 → 102 [ACK] Seq
333	192.168.25.87	192.168.25.200	COTP	102	61162	CR TPDU src-ref: 0x00
334	192.168.25.200	192.168.25.87	TCP	61162	102	102 → 61162 [ACK] Seq
335	192.168.25.200	192.168.25.87	COTP	61162	102	CC TPDU src-ref: 0x80
336	192.168.25.87	192.168.25.200	MMS	102	61162	initiate-RequestPDU
337	192.168.25.200	192.168.25.87	TCP	61162	102	102 → 61162 [ACK] Seq
338	192.168.25.200	192.168.25.87	MMS	61162	102	initiate-ResponsePDU

Fig. 2. Successful MMS communications startup sequence



```
ISO 8327-1 OSI Session Protocol
    SPDU Type: REFUSE (RF) SPDU (12)
    Length: 9
    Transport_Disconnect
        Parameter type: Transport_Disconnect (17)
        Parameter length: 1
        Flags: 0x01, Transport connection
    Version Number
        Parameter type: Version Number (22)
        Parameter length: 1
        Flags: 0x02, Protocol Version 2
    Reason Code
        Parameter type: Reason Code (50)
        Parameter length: 1
        Reason Code: Session Selector unknown (129)
```

Fig. 3. Incorrect session selector response

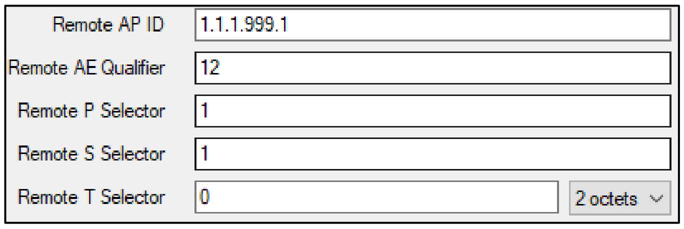


	Source	Destination	Protocol	Destination Port	Source Port	Info
114	192.168.25.87	192.168.25.200	TCP	102	64874	64874 → 102 [SYN] Seq=0
115	192.168.25.200	192.168.25.87	TCP	64874	102	102 → 64874 [SYN, ACK] S
116	192.168.25.87	192.168.25.200	TCP	102	64874	64874 → 102 [ACK] Seq=1
118	192.168.25.87	192.168.25.200	COTP	102	64874	CR TPDU src-ref: 0x0008
119	192.168.25.200	192.168.25.87	TCP	64874	102	102 → 64874 [ACK] Seq=1
120	192.168.25.200	192.168.25.87	COTP	64874	102	CC TPDU src-ref: 0x8001
121	192.168.25.87	192.168.25.200	MMS	102	64874	initiate-RequestPDU
122	192.168.25.200	192.168.25.87	TCP	64874	102	102 → 64874 [ACK] Seq=23
123	192.168.25.200	192.168.25.87	SES	64874	102	REFUSE (RF) SPDU
124	192.168.25.200	192.168.25.87	COTP	64874	102	DR TPDU src-ref: 0x0000
125	192.168.25.87	192.168.25.200	TCP	102	64874	64874 → 102 [ACK] Seq=21
126	192.168.25.87	192.168.25.200	TCP	102	64874	64874 → 102 [FIN, ACK] S
127	192.168.25.200	192.168.25.87	TCP	64874	102	102 → 64874 [ACK] Seq=56

Fig. 4. Communications capture with incorrect COTP parameter

Step 2 includes COTP addressing parameters such as AP ID, AE Qualifier, P Selector, S Selector, and T Selector. These values are populated for both the client and server. Some clients and servers require specific non-default values that can prevent MMS communications from occurring. Some clients and servers may not validate these values, allowing MMS connections to be established with mismatched parameters. A capture analysis can often reveal when these values are incorrect. Fig. 3 shows a case where the session selector (S Selector) parameter was incorrect.

If the TCP socket connection is established, but the connection does not make it to or complete Step 3 (as shown in Fig. 4), then the problem is most likely a mismatch in the COTP parameters. IEDs contain these parameters in their settings or in CID files. Typical COTP values are shown in Fig. 5.

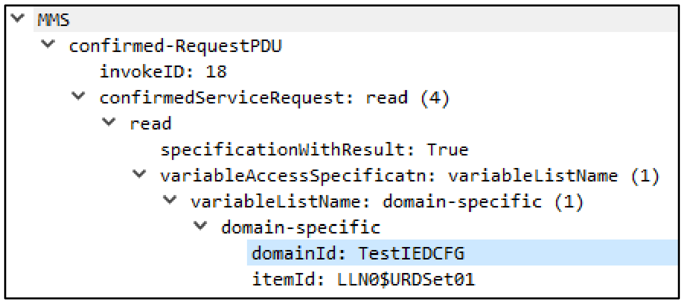


Remote AP ID	1.1.1.999.1	
Remote AE Qualifier	12	
Remote P Selector	1	
Remote S Selector	1	
Remote T Selector	0	2 octets

Fig. 5. Typical COTP values

A common issue that is not necessarily obvious is that the IED name for the server does not match the IED name that the client is configured to use. The IED name is not actually exchanged between the client and server until after the MMS initiate message completes. Another reason this is challenging to diagnose is that there is not a specific error showing that the IED name is incorrect. The server returns the same generic error message as for a report control block, data set, or data object that does not exist.

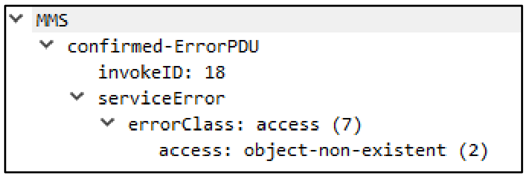
When looking at the network capture, the client will make a request with the IED name and the logical device name concatenated in the domain ID. In the example, as shown in Fig. 6, the IED name is TestIED and the logical device is CFG.



```
MMS
  confirmed-RequestPDU
    invokeID: 18
    confirmedServiceRequest: read (4)
      read
        specificationWithResult: True
        variableAccessSpecificatn: variableListName (1)
          variableListName: domain-specific (1)
            domain-specific
              domainId: TestIEDCFG
              itemId: LLN0$URDSet01
```

Fig. 6. MMS client data set request

The response shown in Fig. 7 contains no additional information other than that the object does not exist. This is the same response given if the logical node name, data object, data set, or report control block do not exist in the IED.

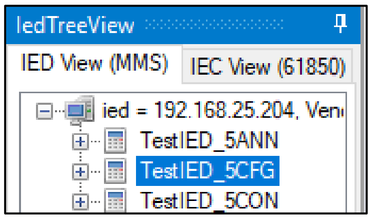


```
MMS
  confirmed-ErrorPDU
    invokeID: 18
    serviceError
      errorClass: access (7)
        access: object-non-existent (2)
```

Fig. 7. MMS server data set response

Since the error message does not describe what is incorrect in the message request, it is best to use an MMS browser to see what is configured in the IED. The MMS browser will ask the IED for a self-description of its data model, which will include the IED name.

The MMS browser used in Fig. 8 concatenates the IED name with the logical device just like in the client request. Depending on the browser used, it may concatenate the IED name and the logical device name or it may use a tree structure with the IED name at the root.



IedTreeView

IED View (MMS) | IEC View (61850)

```
ied = 192.168.25.204, Ven
  TestIED_5ANN
  TestIED_5CFG
  TestIED_5CON
```

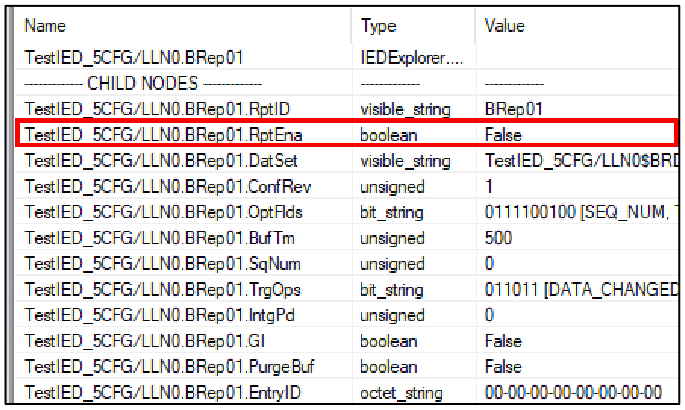
Fig. 8. MMS browser IED name

Once the IED name in the server has been identified, it can be compared against what the client is using and against the CID file that is believed to be active on the server. Resolving a mismatch requires either updating the IEC 61850 configuration in the server to use the correct CID file or updating the client configuration to make requests using the IED name configured in the server.

IV. Report Control Blocks

After a client makes a successful connection to the MMS server, integrators often enable report control blocks to define how the server transmits data to the client. This section of the paper addresses challenges that integrators might face when doing so.

The first step in troubleshooting a report control block that is not connected is to check whether the client or server offers statistics on report control blocks. If they do not, an MMS browser can be used to look at the report control block. The most important attribute to look at in the browser is the report enable attribute (RptEna). If this value is set to true, then a client has activated that report control block. If RptEna is set to false, then that report control block is not enabled and no data transfer is occurring. For example, the report control block in the MMS browser shown in Fig. 9 is not enable, as indicated by the false RptEna value.

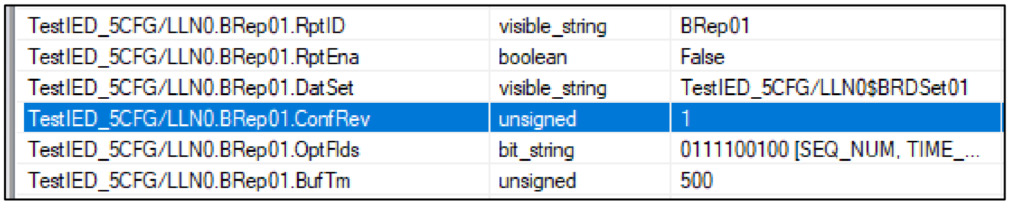


Name	Type	Value
TestIED_5CFG/LLN0.BRep01	IEDExplorer....	
------------ CHILD NODES ------------	------------	------------
TestIED_5CFG/LLN0.BRep01.RptID	visible_string	BRep01
TestIED_5CFG/LLN0.BRep01.RptEna	boolean	False
TestIED_5CFG/LLN0.BRep01.DatSet	visible_string	TestIED_5CFG/LLN0$BRI
TestIED_5CFG/LLN0.BRep01.ConfRev	unsigned	1
TestIED_5CFG/LLN0.BRep01.OptFlds	bit_string	0111100100 [SEQ_NUM, T
TestIED_5CFG/LLN0.BRep01.BufTm	unsigned	500
TestIED_5CFG/LLN0.BRep01.SqNum	unsigned	0
TestIED_5CFG/LLN0.BRep01.TrgOps	bit_string	011011 [DATA_CHANGED
TestIED_5CFG/LLN0.BRep01.IntgPd	unsigned	0
TestIED_5CFG/LLN0.BRep01.GI	boolean	False
TestIED_5CFG/LLN0.BRep01.PurgeBuf	boolean	False
TestIED_5CFG/LLN0.BRep01.EntryID	octet_string	00-00-00-00-00-00-00-00

Fig. 9. MMS browser showing the current status of a report control block

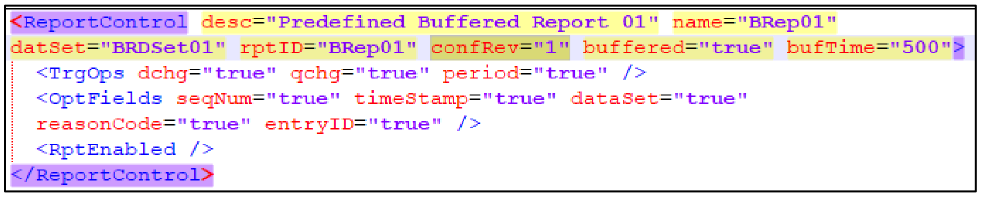
A. Configuration Revision Issues

The IEC 61850 data model uses the configuration revision attribute (ConfRev) in various portions of the report control block configuration to indicate that the configuration has been changed. Each report control block has its own configuration revision attribute. Many clients read the configuration revision of the report control block that is active in the MMS server (as shown in Fig. 10) and then compare it against the configuration revision in the CID file. If the configuration revision matches, the client enables the report control block and starts collecting data.



TestIED_5CFG/LLN0.BRep01.RptID	visible_string	BRep01
TestIED_5CFG/LLN0.BRep01.RptEna	boolean	False
TestIED_5CFG/LLN0.BRep01.DatSet	visible_string	TestIED_5CFG/LLN0$BRDSet01
TestIED_5CFG/LLN0.BRep01.ConfRev	unsigned	1
TestIED_5CFG/LLN0.BRep01.OptFlds	bit_string	0111100100 [SEQ_NUM, TIME_...
TestIED_5CFG/LLN0.BRep01.BufTm	unsigned	500

Fig. 10. MMS browser reading configuration revision of the report control block



```
<ReportControl desc="Predefined Buffered Report 01" name="BRep01"
datSet="BRDSet01" rptID="BRep01" confRev="1" buffered="true" bufTime="500">
  <TrgOps dchg="true" qchg="true" period="true" />
  <OptFields seqNum="true" timeStamp="true" dataSet="true"
  reasonCode="true" entryID="true" />
  <RptEnabled />
</ReportControl>
```

Fig. 11. Report control block configuration revision in CID file

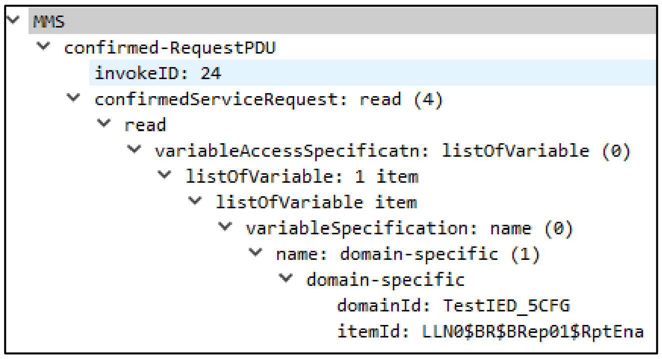
If the report control block configuration revision does not match what is listed in the CID file (Fig. 11), the client may not enable the report control block. Either the client or server must be updated to ensure that the configuration revisions match. The reason the configuration revisions may not match is because software applications commonly increment this value after a settings change related to the report control block.

B. *Another Client Already Connected to Control Block and Indexing Functionality*

It is common in substations for multiple clients to collect the same information from a single IED. Sometimes a client cannot enable a report control block but can connect to the MMS server and collect other information. It is important to check whether another client already has an active connection to a report control block. The best way to troubleshoot this is to take a communications capture.

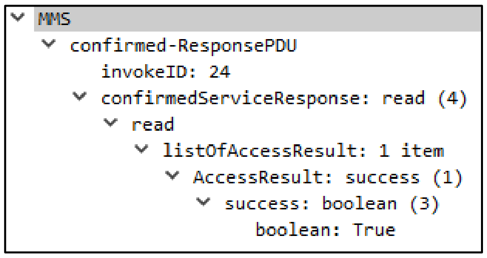
Most MMS clients read the RptEna attribute in the server prior to attempting to write to it. Communications captures, like those shown in Fig. 12 and Fig. 13, can show whether the RptEna attribute is already set to true.

When RptEna is set to true, the client already connected to the report control block needs to be identified. Some servers offer statistics about the connected client, such as its IP address, but this is not common. If the connected client is configured incorrectly, then it needs to be removed or disabled so that the intended client can connect to the server's report control block.



```
MMS
  confirmed-RequestPDU
    invokeID: 24
    confirmedServiceRequest: read (4)
      read
        variableAccessSpecificatn: listOfVariable (0)
          listOfVariable: 1 item
            listOfVariable item
              variableSpecification: name (0)
                name: domain-specific (1)
                  domain-specific
                    domainId: TestIED_5CFG
                    itemId: LLN0$BR$BRep01$RptEna
```

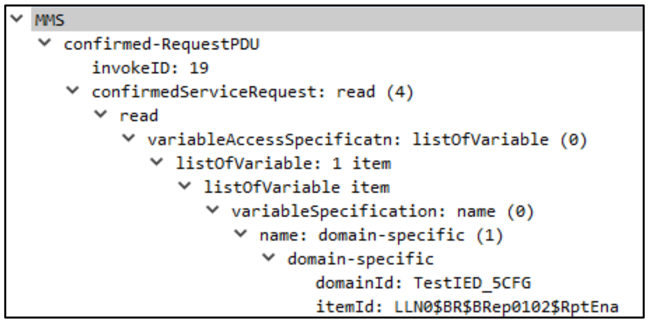
Fig. 12. MMS client read of RptEna



```
MMS
  confirmed-ResponsePDU
    invokeID: 24
    confirmedServiceResponse: read (4)
      read
        listOfAccessResult: 1 item
          AccessResult: success (1)
            success: boolean (3)
              boolean: True
```

Fig. 13. MMS server RptEna response

Some servers can support multiple clients connecting to the same report control block. This feature, described by the IEC 61850 standard, is called indexing [4]. Not all IEDs support this feature. Indexing allows clients to add an index or instance number to the report control block name. In the examples used in this section, the report control block is named BRep01, short for "Buffered Report 01." To ask for the first or second instance of that report control block, the client would ask for BRep0101 or BRep0102, respectively (see Fig. 14).



```
MMS
  confirmed-RequestPDU
    invokeID: 19
    confirmedServiceRequest: read (4)
      read
        variableAccessSpecificatn: listOfVariable (0)
          listOfVariable: 1 item
            listOfVariable item
              variableSpecification: name (0)
                name: domain-specific (1)
                  domain-specific
                    domainId: TestIED_5CFG
                    itemId: LLN0$BR$BRep0102$RptEna
```

Fig. 14. Request for second instance of the report control block

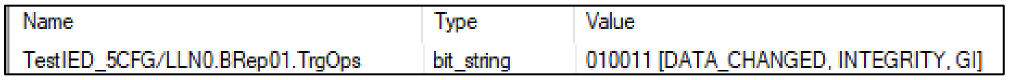
The configuration of indexes for report control blocks varies across MMS clients from different manufacturers. Some clients support a search mode in which the client asks for the report control block instance. If RptEna is set to true for that instance, the client increments the index and attempts to read RptEna until it finds an available index for that report control block.

While this type of behavior makes it easy to configure the client, it presents some challenges. For buffered reports, which keep track of the entryID of a client that disconnects from one instance and then connects to another instance, the report control block may not have the same entryID values or they may be purged, resulting in data loss. It is better to configure each client to use a specific index of that report control block to prevent conflicts between clients.

C. *Trigger Options*

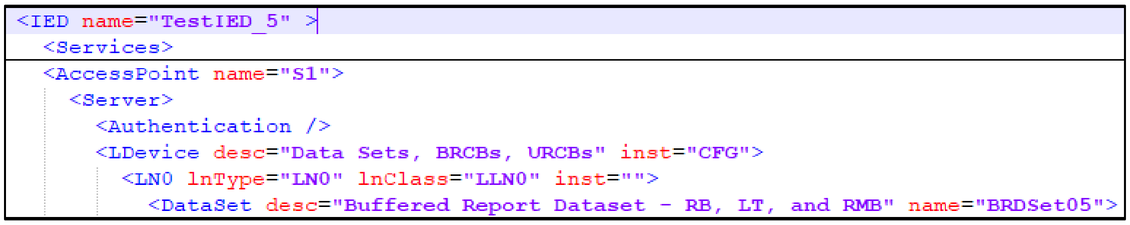
Another integration challenge occurs when the report control block is connected and has RptEna enabled, but either no data are sent, event data are not sent, or quality changes are not sent. This occurs because report control blocks have a configuration parameter called trigger options [5]. These options are encoded in a six-bit string as follows:

• Bit 0	Reserved
• Bit 1	Data change
• Bit 2	Quality change
• Bit 3	Data update
• Bit 4	Integrity
• Bit 5	General interrogation



Name	Type	Value
TestIED_5CFG/LLN0.BRep01.TrgOps	bit_string	010011 [DATA_CHANGED, INTEGRITY, GI]

Fig. 15. MMS browser trigger options



```
<IED name="TestIED_5" >
  <Services>
  <AccessPoint name="S1">
    <Server>
      <Authentication />
      <LDevice desc="Data Sets, BRCBs, URCBs" inst="CFG">
        <LN0 lnType="LN0" lnClass="LLN0" inst="">
          <DataSet desc="Buffered Report Dataset - RB, LT, and RMB" name="BRDSet05">
```

Fig. 16. Data set example in SCL

These trigger options tell the report control block when to send data to the client. The client initiates the general interrogation trigger option to receive the present status of the data set monitored by the report control block. The integrity trigger option, when enabled, tells the server to periodically send the client the present status of all items in the data set regardless of any changes. The data change, quality change, and data update trigger options determine during runtime when changes occur in the data set and send those changes to the client.

If the client only periodically receives updates from the server and not as changes occur, it is likely that the report control block has the integrity trigger option enabled but does not have the data change trigger option enabled. The IEC 61850 standard does not specify if it is the client's or server's responsibility to determine which trigger options to enable. This leaves it to implementers to make sure the correct trigger options are enabled. It is common for servers to offer settings to determine which trigger options are enabled by default. When clients enable a report control block, they may attempt to enable all trigger options to ensure that data are transmitted. Even when clients attempt to enable trigger options, however, servers sometimes do not support writing to certain trigger options, which causes the trigger option write to fail and prevents the IED from transmitting data when the report control block is enabled.

To determine what trigger options are enabled in a report control block, an MMS browser is the best tool. Most MMS browsers can decode the bit string that stores the trigger options. However, trigger options are not always enumerated, so it is important to be able to decode the bit string manually.

Fig. 15 shows the trigger options encoded in the bit string. The bit string is read from left to right [5]. Bit 0 is on the left side and is always 0 because it is reserved. If the desired trigger options are not enabled, it is best to enable them in the server settings or the CID file.

D. *Data Do Not Update Correctly*

Another challenge when integrating IEC 61850 MMS communications is that after the socket connection has been established and the data set or report control blocks return information, the data may still not update correctly. Identifying this challenge is more difficult than identifying the other challenges discussed in this paper, but all of them are ultimately the result of a difference in the IEC 61850 model between the client and server that requires one to be updated to match the other.

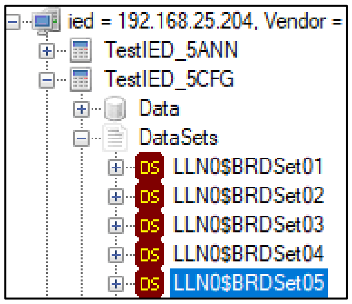
1) Data Set or Data Object Does Not Exist

The most common challenge in this category occurs when the server has a different configuration than the client and the server does not contain the data sets that were configured in its software. In Fig. 7, the incorrect name error response from the IED was shown. Unfortunately, when a data set, report control block, or data object does not exist, the same error message results: object-non-existent.

In this situation, an Ethernet packet capture shows that the IED does not contain the requested information. However, it does not show what information the IED actually has. The best tool to discover this is an MMS browser. To verify whether an IED has a data set or report control block, it is easiest to look at the CID file to see where that data set or control block lives in the data model. Most of the time, data sets and report control blocks live in the logical node LLN0 in each logical device. But they are not required to be in that location [6].

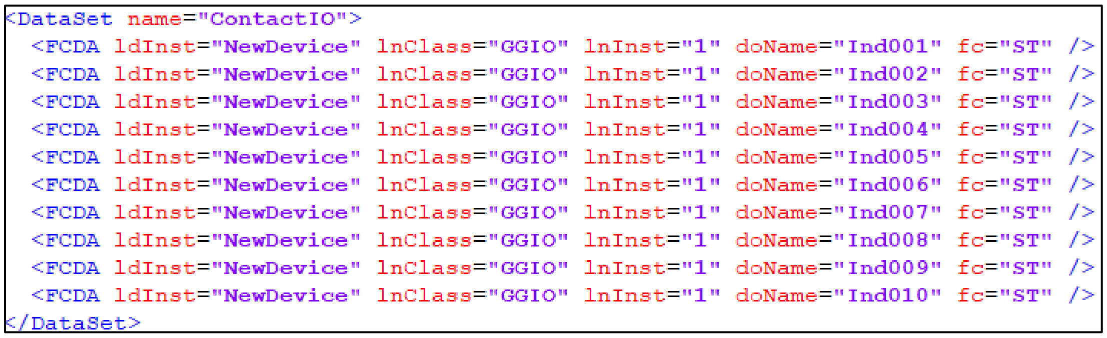
When navigating through any IEC 61850 SCL file, like that shown in Fig. 16, many people find it easier to use a tool that supports XML formatting because the elements that make up the IEC 61850 configuration file are collapsible. This makes navigating the file and finding the desired information significantly easier.

For example, looking for the data set BRDSet05 in the IED data model shows that it should exist in the logical device CFG in the logical node LLN0. Fig. 17 shows the corresponding location in the MMS browser.



```
ied = 192.168.25.204, Vendor =
  TestIED_5ANN
  TestIED_5CFG
    Data
    DataSets
      DS LLN0$BRDSet01
      DS LLN0$BRDSet02
      DS LLN0$BRDSet03
      DS LLN0$BRDSet04
      DS LLN0$BRDSet05
```

Fig. 17. Data set example in MMS browser



```
<DataSet name="ContactIO">
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind001" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind002" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind003" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind004" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind005" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind006" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind007" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind008" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind009" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind010" fc="ST" />
</DataSet>
```

Fig. 18. Example binary data set

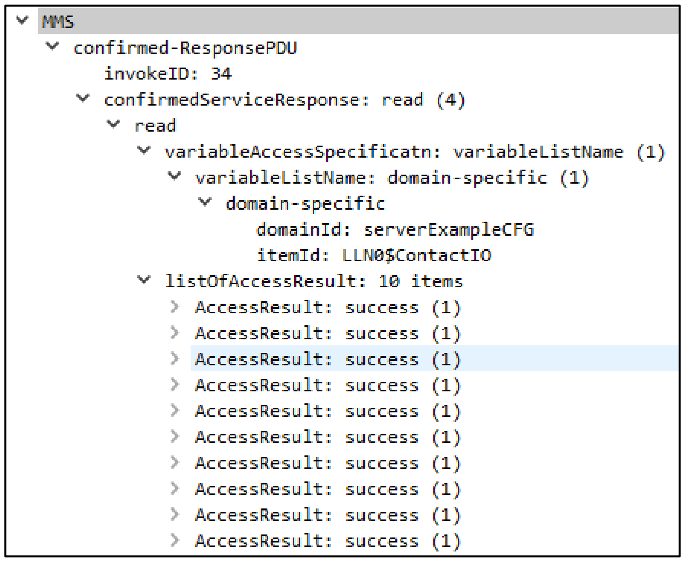
In this MMS browser, the data sets are separated into easy-to-find locations for each logical device. This kind of feature is why using an MMS browser makes troubleshooting easier than a more traditional MMS client-and-server communication session. It provides the ability to see what the server has configured rather than relying on a settings file or CID file that (depending on the manufacturer) may be difficult to verify.

2) *Data Set in the IED Does Not Match*

In the MMS protocol, when a client asks for a data set and the server transmits the data back to the client, all of the data are ANSI-encoded and contain no naming information regarding what the data pertain to in the data model. This means that the client must know prior to asking for a data set how the items returned in the data set correspond to the IEC 61850 data model in the IED. When the configurations in the client and server are the same, everything lines up and there are no problems. However, when the configurations are different, the content of the data set (i.e., the information sent to the client) no longer matches what is in the client. The client is in a difficult position. It does its best to process the data. Many clients process as much of the data as possible and stop updating items in the data set if a non-matching data type is found. The result is that some data from the IED update and other data do not.

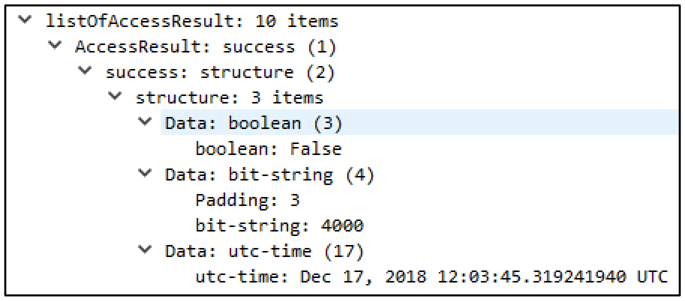
Fig. 18 shows a data set that contains ten single point status (SPS) points. When the client asks for data set "ContactIO," it receives ten SPS points without identifying tag names, as shown in Fig. 19. The response contains the data set name and each of the SPS points as AccessResult items.

Fig. 19 shows a detailed view of one of the SPS points that was returned. There is no descriptive naming; it simply contains a Boolean status value, a quality (which is returned as a bit string), and a time stamp. These are the components that make up a status (ST), functionally constrained SPS point. In the response shown in Fig. 19, each of the AccessResult items has the detailed view like that shown in Fig. 20.



```
MMS
  confirmed-ResponsePDU
    invokeID: 34
    confirmedServiceResponse: read (4)
      read
        variableAccessSpecificatn: variableListName (1)
          variableListName: domain-specific (1)
            domain-specific
              domainId: serverExampleCFG
              itemId: LLN0$ContactIO
        listOfAccessResult: 10 items
          AccessResult: success (1)
          AccessResult: success (1)
          AccessResult: success (1)
          AccessResult: success (1)
          AccessResult: success (1)
          AccessResult: success (1)
          AccessResult: success (1)
          AccessResult: success (1)
          AccessResult: success (1)
          AccessResult: success (1)
```

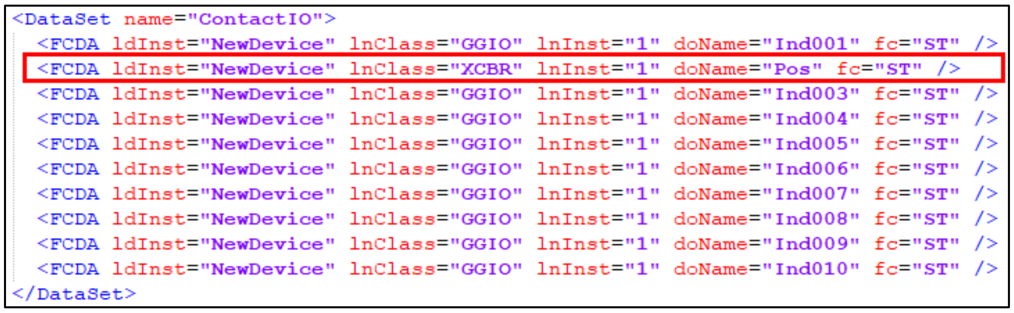
Fig. 19. ContactIO data set response



```
listOfAccessResult: 10 items
  AccessResult: success (1)
    success: structure (2)
      structure: 3 items
        Data: boolean (3)
          boolean: False
        Data: bit-string (4)
          Padding: 3
          bit-string: 4000
        Data: utc-time (17)
          utc-time: Dec 17, 2018 12:03:45.319241940 UTC
```

Fig. 20. Detailed SPS view in data set response

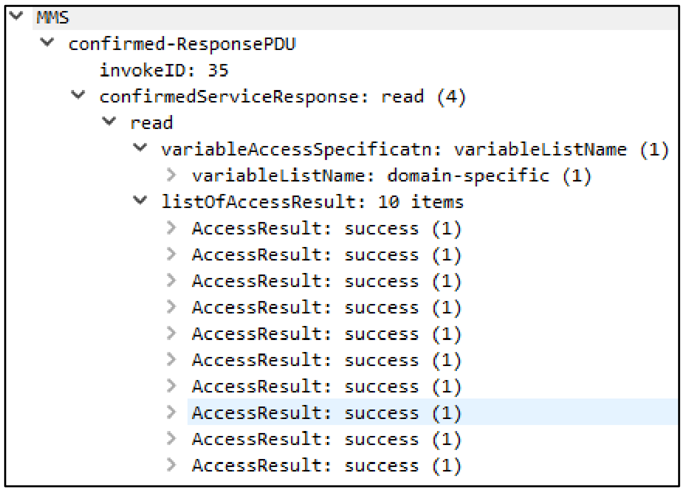
However, if something is changed in the data set (e.g., an SPS point is replaced with a double point status [DPS] point), the change is obvious when looking at the data set definition, as shown in Fig. 21.



```
<DataSet name="ContactIO">
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind001" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="XCBR" lnInst="1" doName="Pos" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind003" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind004" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind005" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind006" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind007" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind008" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind009" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind010" fc="ST" />
</DataSet>
```

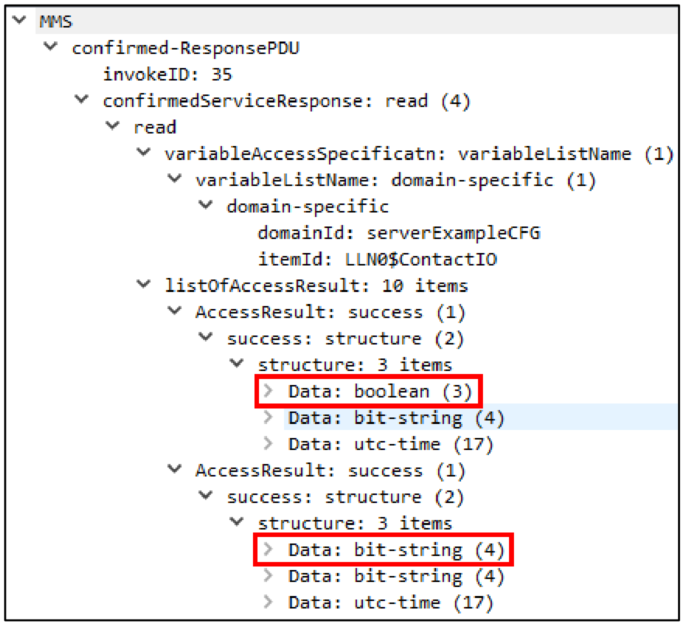
Fig. 21. ContactIO data set with DPS point

The Wireshark capture in Fig. 22 at the top level looks identical to Fig. 19. However, the detailed view in Fig. 23 shows that the second item in the list is not an SPS type. The primary marker identifying this difference is the bit-string status type of the DPS point rather than the Boolean status type of the SPS point.



```
MMS
  confirmed-ResponsePDU
    invokeID: 35
    confirmedServiceResponse: read (4)
      read
        variableAccessSpecificatn: variableListName (1)
          variableListName: domain-specific (1)
        listOfAccessResult: 10 items
          AccessResult: success (1)
          AccessResult: success (1)
          AccessResult: success (1)
          AccessResult: success (1)
          AccessResult: success (1)
          AccessResult: success (1)
          AccessResult: success (1)
          AccessResult: success (1)
          AccessResult: success (1)
          AccessResult: success (1)
```

Fig. 22. ContactIO data set response with DPS point



```
MMS
  confirmed-ResponsePDU
    invokeID: 35
    confirmedServiceResponse: read (4)
      read
        variableAccessSpecificatn: variableListName (1)
          variableListName: domain-specific (1)
            domain-specific
              domainId: serverExampleCFG
              itemId: LLN0$ContactIO
        listOfAccessResult: 10 items
          AccessResult: success (1)
            success: structure (2)
              structure: 3 items
                Data: boolean (3)
                Data: bit-string (4)
                Data: utc-time (17)
          AccessResult: success (1)
            success: structure (2)
              structure: 3 items
                Data: bit-string (4)
                Data: bit-string (4)
                Data: utc-time (17)
```

Fig. 23. Detailed DPS view in data set response

When troubleshooting, it is common to have only the client configuration and the communications capture. Using these two items, the method discussed in this section can be used to determine that the client data set configuration does not match the server configuration. Analyzing this situation requires first comparing the number of items that appear under the AccessResult list in the capture to the number of functionally constrained data attribute (FCDA) items in the data set in the SCL file. If the two do not match, then the data transfer between the client and server will not work completely. As mentioned previously, most clients process the data that they are able to from the server response, which allows some percentage of data to update correctly.

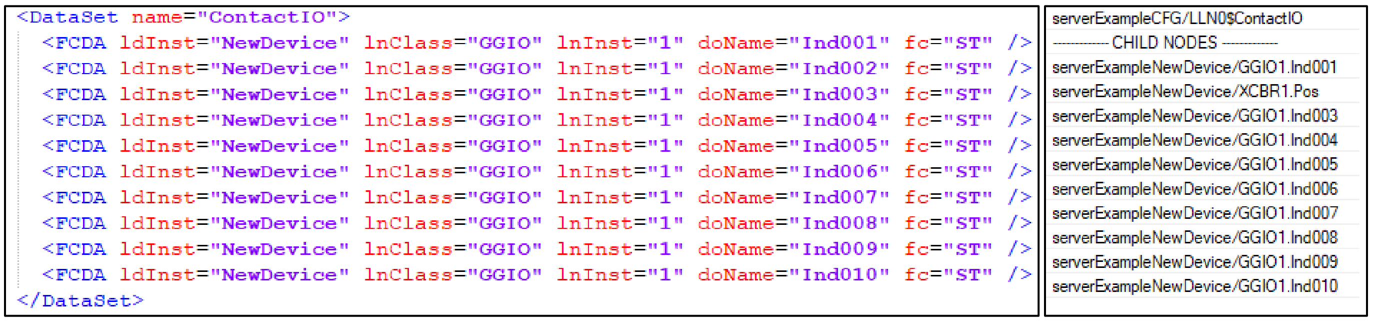
If an MMS browser is available, then comparing the self-discovered model to the client configuration also highlights any discrepancies between the client and server. The difference is in the data object name, as shown in Fig. 24. The resolution to this particular issue is to update the client and server to work with the same CID file.

E. Duplicate Events From Report Control Blocks

Most common SCADA communications protocols (e.g., DNP3, IEC 60870-5-101/104, IEC 61850 MMS) have many features in common, including general features, polling data, unsolicited reporting, controls, and event buffering. In most of these protocols, it is the server's responsibility to keep track of which events have been transmitted to the client. Once the server transmits the event and receives an acknowledgement from the client, the server clears that event from its buffer.

However, this is not true for the IEC 61850 standard. The server instead buffers events in the buffered report control block and assigns an entryID to each report with events [4]. The server sends this entryID to the client with each report. It is then the client's responsibility to keep track of which entryIDs it has already processed.

In Edition 1 of IEC 61850, most servers were required to transmit the last report that had not been sent to the client already. If a client was connected and processing a buffered report control block and then lost communications, the server would buffer changes. Once the client reconnected, the server would start transmitting reports that it had not sent the client the last time the client was connected. While not exactly the same, this behavior was similar to that of the DNP3 and IEC 60870-5-101/104 SCADA protocols, where the server keeps track of event buffering. It also allowed the client to specify an earlier entryID if it wanted older reports.



```
<DataSet name="ContactIO">
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind001" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind002" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind003" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind004" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind005" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind006" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind007" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind008" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind009" fc="ST" />
  <FCDA ldInst="NewDevice" lnClass="GGIO" lnInst="1" doName="Ind010" fc="ST" />
</DataSet>
```

```
serverExampleCFG/LLN0$ContactIO
------------ CHILD NODES ------------
serverExampleNewDevice/GGIO1.Ind001
serverExampleNewDevice/XCBR1.Pos
serverExampleNewDevice/GGIO1.Ind003
serverExampleNewDevice/GGIO1.Ind004
serverExampleNewDevice/GGIO1.Ind005
serverExampleNewDevice/GGIO1.Ind006
serverExampleNewDevice/GGIO1.Ind007
serverExampleNewDevice/GGIO1.Ind008
serverExampleNewDevice/GGIO1.Ind009
serverExampleNewDevice/GGIO1.Ind010
```

Fig. 24. Data set in SCL versus data set in MMS browser

In Edition 2 of IEC 61850, the server behavior has an extra definition added. When an MMS client connects to a buffered report control block, if the client has not written to the entryID in the MMS server prior to RptEna being set to true, then the server is required to transmit all the reports in its memory [4]. This places a greater responsibility on the client to keep track of which reports have and have not been processed. Since this is a change in how clients typically process data, some clients may not be able to handle this situation. If clients do not have active management of the entryID, then it is possible for them to process events from servers multiple times if the client power-cycles or loses communications. If there are duplicate events in a sequence of events log, this is likely the situation that is occurring. This issue can be resolved by checking with the manufacturer of the client and server to see what options are available for managing the entryID in buffered report control blocks.

V. Control Services

Like other SCADA protocols, IEC 61850 offers control services, but it also offers more verification and configuration capabilities for controls than other SCADA protocols do. However, this provides more opportunities for integration challenges rather than enhanced functionality or more productive systems. Unlike troubleshooting data transfer in MMS, where error messages are very generic, control services generally give specific error messages that allow for easier resolution once the messages are identified.

A. Control Models

The IEC 61850 standard has five different control models. However, there are only two types of controls with two modes each. The fifth model is "status only," which indicates that the control data type is not controllable.

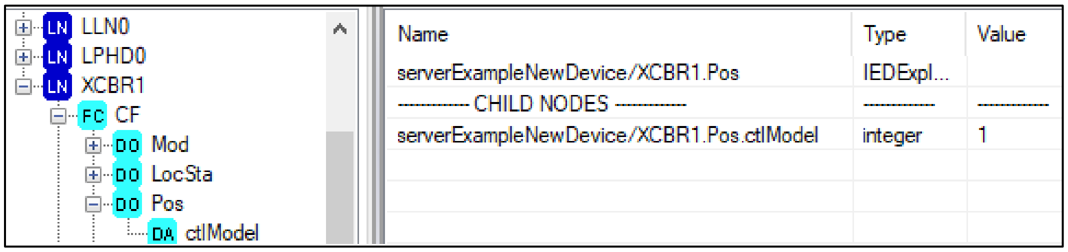
The two modes are called normal security and enhanced security. The naming of these modes is not intuitive. These two modes primarily drive client-side statistics related to whether a control is considered successful. When a control is sent from the client to the server, the server can either accept or reject that message. When the control model is normal security, this response drives the client's statistics as to whether the control failed or was successful [4].

When the control model is enhanced security, the client determines if the control is successful based on additional information. After the server accepts the control message, it attempts to let that control make a change in the logic engine, and the server expects the status portion of that control to change as a result within a certain timeframe. After the time period (which is configurable in the server) has passed, the server sends an unsolicited message to the client, which is called the last application report. This message includes information about the control that the client sent to the server and whether the server was able to operate the control and have a corresponding status change. If the status did not change as expected, then the last application report will contain an error message and the client will mark that control as failed [4].

The two types of controls are direct operate and select-before-operate. The difference between the two is that in select-before-operate, a select message is sent prior to the operate message [4].

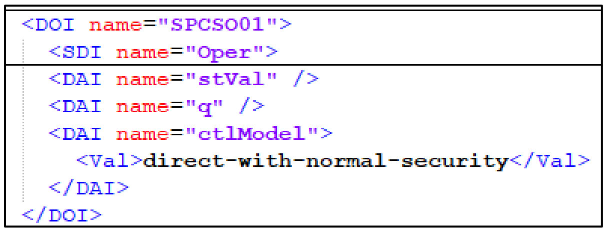
The most important key to successful control operations is that the control model match between the client and server. Depending on the client being implemented, the steps required to match the control models varies. Some clients simply read the control model that the server has active and use that model when issuing controls to the server. Some clients use the model defined in the CID file, and some expect a user-entered control model for the configuration.

When troubleshooting controls, it is important to make sure that the control model is set to a controllable value. There are IEDs that have XCBR and XCWI logical nodes with data objects that have a control-based common data class (CDC) type but with the control model set to status only. The fact that these controls are set to status only is not always intuitive based on the IEC 61850 data model for controlling circuit breakers or circuit switches. In these cases, the instruction manual for the IED or the configuration should be consulted to see how logic engine points are mapped to the IEC 61850 data model.



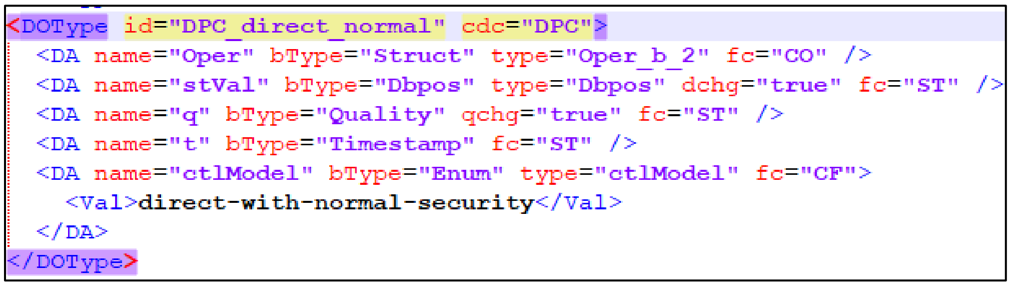
Name	Type	Value
serverExampleNewDevice/XCBR1.Pos	IEDExpl...	
------------ CHILD NODES ------------	------------	------------
serverExampleNewDevice/XCBR1.Pos.ctlModel	integer	1

Fig. 25. MMS browser reading of control model



```
<DOI name="SPCSO01">
  <SDI name="Oper">
  <DAI name="stVal" />
  <DAI name="q" />
  <DAI name="ctlModel">
    <Val>direct-with-normal-security</Val>
  </DAI>
</DOI>
```

Fig. 26. Control model definition in data object instantiation



```
<DOType id="DPC_direct_normal" cdc="DPC">
  <DA name="Oper" bType="Struct" type="Oper_b_2" fc="CO" />
  <DA name="stVal" bType="Dbpos" type="Dbpos" dchg="true" fc="ST" />
  <DA name="q" bType="Quality" qchg="true" fc="ST" />
  <DA name="t" bType="Timestamp" fc="ST" />
  <DA name="ctlModel" bType="Enum" type="ctlModel" fc="CF">
    <Val>direct-with-normal-security</Val>
  </DA>
</DOType>
```

Fig. 27. Control model definition in data object type definition

To see what control model is in use, the easiest method is to use an MMS browser and see what the self-discovery method shows. If the MMS browser has data sorted by functional constraint, it is important to look under the CF constraint to see the control model, as shown in Fig. 25. The control models correlate to following values [5]:

- 0 = status-only
- 1 = direct-with-normal-security
- 2 = sbo-with-normal-security
- 3 = direct-with-enhanced-security
- 4 = sbo-with-enhanced-security

In the CID file, the control model can be placed in the data object instantiation for the control point or in the data object type definition. The data object instantiation for the control point [6] is in the server element in the SCL file. Each control point in the IEC 61850 data model has the control model defined, as shown in Fig. 26. If there are many controllable objects in the data model, it will enlarge the CID file. And, since the control models are likely the same, the information is redundant.

To help reduce the redundancy of the control model information, the IEC 61850 standard also allows the control model definition to be placed in the data object type definition. This means that, for all data objects of a particular control type that reference the data object type definition, the control model is listed only once [6], reducing the size of the SCL file. An example of this is shown in Fig. 27.

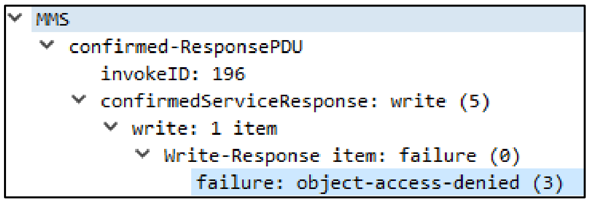
B. *orCat Not Supported*

IEC 61850 controls support the concept of control origin identity. With each control operation, an enumeration is sent from the client to the server that provides a description of the client device. This enumeration is defined as the orCat attribute, which is an abbreviation for originator category [5].

These attributes are defined as follows:

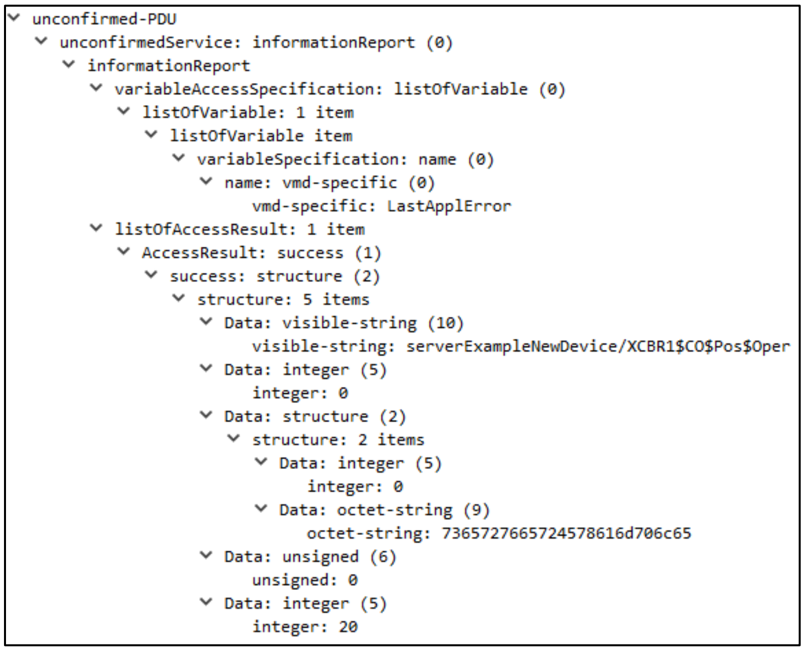
- 0 = not-supported
- 1 = bay-control
- 2 = station-control
- 3 = remote-control
- 4 = automatic-bay
- 5 = automatic-station
- 6 = automatic-remote
- 7 = maintenance
- 8 = process

Some servers only support specific orCat values when a control is transmitted, and some servers support configurable orCat values. For example, a server may be configured to only accept controls that originate from 3 (remote-control) or 6 (automatic-remote). Alternatively, the IED could be put into a test or local mode for maintenance and only accept controls that have orCat values of 2 (station-control) or 7 (maintenance).



```
MMS
  confirmed-ResponsePDU
    invokeID: 196
    confirmedServiceResponse: write (5)
      write: 1 item
        Write-Response item: failure (0)
          failure: object-access-denied (3)
```

Fig. 28. MMS control failure due to unsupported orCat



```
unconfirmed-PDU
  unconfirmedService: informationReport (0)
    informationReport
      variableAccessSpecification: listOfVariable (0)
        listOfVariable: 1 item
          listOfVariable item
            variableSpecification: name (0)
              name: vmd-specific (0)
                vmd-specific: LastApplError
      listOfAccessResult: 1 item
        AccessResult: success (1)
          success: structure (2)
            structure: 5 items
              Data: visible-string (10)
                visible-string: serverExampleNewDevice/XCBR1$CO$Pos$Oper
              Data: integer (5)
                integer: 0
              Data: structure (2)
                structure: 2 items
                  Data: integer (5)
                    integer: 0
                  Data: octet-string (9)
                    octet-string: 7365727665724578616d706c65
              Data: unsigned (6)
                unsigned: 0
              Data: integer (5)
                integer: 20
```

Fig. 29. Last application report for unsupported orCat value

If a server does not support an orCat value that the client sends, the server will respond to the MMS write command with the MMS error shown in Fig. 28 [5]. In addition, the server will likely send a last application report. The last application report sends the information in Table III. The last value in the last application report is AddCause, which shows a more specific error. In Fig. 29, the AddCause is 20, which correlates to an enumeration definition indicating that the error message has no access authority.

TABLE III
LAST APPLICATION ERROR MESSAGE STRUCTURE

Component Name	Type Description	Read/ Write (r/w)	Mandatory/ Optional (m/o)	Comments
CntrlObj	VISIBLESTRING	r	m	–
Error	ENUMERATED	r	m	–
Origin	Originator	r	m	See IEC 61850-7-3
ctlNum	INT8U	r	m	See IEC 61850-7-3
AddCause	ENUMERATED	r	m	–

Following is a list of all other possible AddCause values:

- 0 = Unknown
- 1 = Not-supported
- 2 = Blocked-by-switching-hierarchy
- 3 = Select-failed
- 4 = Invalid-position
- 5 = Position-reached
- 6 = Parameter-change-in-execution
- 7 = Step-limit
- 8 = Blocked-by-mode
- 9 = Blocked-by-process
- 10 = Blocked-by-interlocking
- 11 = Blocked-by-synchrocheck
- 12 = Command-already-in-execution
- 13 = Blocked-by-health
- 14 = 1-of-n-control
- 15 = Abortion-by-cancel
- 16 = Time-limit-over
- 17 = Abortion-by-trip
- 18 = Object-not-selected
- 19 = Object-already-selected
- 20 = No-access-authority
- 21 = Ended-with-overshoot
- 22 = Abortion-due-to-deviation
- 23 = Abortion-by-communication-loss
- 24 = Blocked-by-command
- 25 = None
- 26 = Inconsistent-parameters
- 27 = Locked-by-other-client

The last application report error message can be descriptive. For example, position-reached indicates that a control was sent for which the status already matched the value sent. Blocked-by-switching-hierarchy indicates that the device is in a local mode. Other AddCause values are less descriptive, such as time-limit-over. This AddCause value indicates that the control was sent to and accepted by the server but the status portion of the control structure did not change. This indicates that the status value corresponding to the control may not be mapped to the status of the control objection in the IED logic engine. But it does not definitively answer why the control did not succeed.

VI. Conclusion

This paper covers a range of integration challenges that users implementing IEC 61850 solutions may encounter. The resolution for all of these issues ultimately is to make sure that the client and server are operating on the same IEC 61850 configuration. While this resolution may seem obvious, many IEC 61850 server configurations are loaded via a manufacturer's settings file rather than being created with an SCL file. It is not always obvious whether an IED has the latest configuration active on it. Often, the active configuration has parts of the client-and-server communications working, which makes troubleshooting less intuitive. This paper covers how to identify, troubleshoot, and resolve problem behaviors.

VII. References

[1] Wireshark, "Internet (TCP/IP) Protocol Family," September 2014. Available: https://wiki.wireshark.org/InternetProtocolFamily.

[2] Wireshark, "ISO Protocol Family," August 2013. Available: https://wiki.wireshark.org/IsoProtocolFamily.

[3] Wireshark, "Connection Oriented Transport Protocol (COTP, ISO 8073)," April 2008. Available: https://wiki.wireshark.org/COTP.

[4] IEC 61850-7-2 Edition 2, Communication networks and systems for power utility automation – Part 7-2: Basic information and communication structure – Abstract communication service interface (ACSI), 2010.

[5] IEC 61850-8-1, Communication networks and systems for power utility automation – Part 8-1: Specific communication service mapping (SCSM) – Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3, 2011.

[6] IEC 61850-6, Communication networks and systems for power utility automation – Part 6: Configuration description language for communication in electrical substations related to IEDs, 2009.

VIII. Biographies

Kevin Mahoney is the president and founder of Casco Systems, LLC, in Cumberland, Maine. He has over 30 years of experience in control and system integration engineering. He has led and worked on many integration and control projects, including 345/115 kV bulk power substations, wind farm substations, hydro station automation upgrades, and many SCADA systems. These projects included communications design, IEC 61850 GOOSE relay-to-relay communications, human-machine interfaces, and system commissioning. In 2017, he received the IEEE PES Maine Outstanding Engineer Award. Kevin graduated from the University of New Hampshire with a B.S. degree in electrical engineering technology.

Kevin Coyne is a technical manager at Casco Systems, LLC, in Cumberland, Maine. He has over 25 years of experience designing, programming, and commissioning automated control systems. He has worked on multiple 345/115 kV bulk power substations, wind farms, and hydroelectric integration projects. He has experience designing, programming, and commissioning new platforms for clients using IEC 61850 GOOSE for relay-to-relay communications and IEC 61850 MMS for local human-machine interface and remote terminal unit supervisory communications. Kevin graduated from University of Maine with a B.S. degree in electrical engineering technology.

Brian Waldron is a lead automation engineer with Schweitzer Engineering Laboratories, Inc. (SEL) in Pullman, Washington. He has several years of experience in designing and troubleshooting automation systems and communications networks. He has authored several application guides focusing on integrating automation products. He has represented SEL at IEC 61850 interoperability demonstrations organized by Utility Communications Architecture (UCA) and frequently teaches engineering design and the application of IEC 61850 solutions. Brian graduated from Gonzaga University with a B.S. degree in electrical engineering.