IEC 61968 – MULTISPEAK® HARMONIZATION

Gerald GRAY
Guiding Principle Consulting – United States
gerald.gray@guiding-principle.com

John SIMMINS
EPRI – United States
jsimmins@epri.com

Gary MCNAUGHTON
Cornice Engineering – United States
gmcnaughton@corniceengineering.com

David HAYNES
Aclara – United States
DHaynes@aclara.com

Robert SAINT
NRECA – United States
Robert.saint@nreca.coop

Alan McMorran
Open Grid Systems, Ltd – United Kingdom
alan@mcmorran.co.uk

Eric LAMBERT
EDF – France
Eric.lambert@edf.fr

ABSTRACT

Multiple standards that cover the same information domain present a problem for the vendor community when developing products to serve the industry. The classic challenge becomes one of determining which standards to support or how best to support one or both standards. The classic harmonization solution involves performing a mapping from one standard to another, which while useful, (and which is also covered here) only addresses the problem at a given point in time. However, looking forward and expressing each standard as an ontology gives the user a deeper understanding of the information that is exposed and also facilitates the understanding of relationships between classes that may not have been readily apparent previously. In addition to examining the mapping between CIM and MultiSpeak®, the process of converting artifacts from the two standards into a formal web ontology language (OWL) is also covered.

INTRODUCTION

At present there are two leading standards for software interoperability in the electric utility industry: MultiSpeak® and the Common Information Model (CIM), which is maintained by Technical Committee 57 (TC57) of the International Electrotechnical Commission (IEC). The two standards have been developed by different committees that have some overlap in membership. As a result, the two standards cover much of the same material, but in different manners. Multiple standards are important to the creation of the smart grid [1], and standards are generally optimized along a given direction due to the differing needs of the respective stakeholders. Nevertheless, the existence of two somewhat dissimilar standards creates a dilemma for both utilities and vendors. Utilities must decide which of the two standards to support and, in some cases, must interface between two products, each designed for compatibility with different standards. In some cases vendors must support both standards. Similarly, support for both standards lengthens product development times, thus delaying the availability of systems for utility implementation and eventually customer availability.

Previous work on harmonization has been done [2] but this primarily focused on the mapping at the logical level, with an examination of use cases and actors as part of the AMI Enterprise Task Force of the Open Smart Grid effort.

This presentation will report on a mapping between the recently released Version 4.1 of MultiSpeak® and IEC 61968/61970 CIM [8]. This interoperability mapping (i) assists utilities to specify either standard with confidence, knowing that MultiSpeak systems could interoperate with other systems with minimal customization, (ii) permits vendors to develop interfaces that work with systems compatible with either of the standards, and (iii) enables smart grid solutions to be delivered to the public.

THE POINT-TO-POINT PROBLEM

When faced with how to harmonize two different standards that cover the same domain one typically looks at how to map from one class or set of attributes to the classes and attributes presented in the other standard. While having two products separated by competing standards is a problem, having a product based on a completely proprietary interface is worse. With a proprietary interface there is no hope of a future resolution, the interface will always be a custom solution, which, all things being equal, increases maintenance costs.

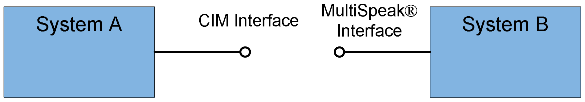


Figure 1: The classic gap in a point-to-point solution when vendors support differing standards

With the situation seen in Figure 1, the organization will typically need to buy systems that support the chosen interface (unlikely in the heterogeneous landscapes of most enterprises), or do a data transformation to support a given interface.

An improvement over the situation depicted in Figure 1 is to utilize an ESB between the differing systems. ESBs offer many capabilities which make them popular in the enterprise; the most significant for the CIM-MultiSpeak problem is the ability of the ESB to map the data of any given interface and reuse that mapping in other integrations. The organization needs only determine how to connect any given system to the ESB and then this interface can be reused across the organization. From an interoperability standpoint the challenge is not so much the interface from points A and B or C and D as seen in Figure 2, as each interface to the ESB can rely on the expected return results of its respective standard for both errors and normal return operations. The greater challenge is when a request has been made say, from System A and then mapping the data for acceptance by System B and also determining how to handle errors that occur at point D in this example and returning them to the calling application in System A.

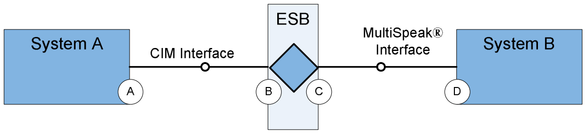


Figure 2: System diagram illustrating how an ESB can be utilized to map between CIM-based or MultiSpeak-based interfaces, promoting reuse

Assumptions

For the ESB discussion the main assumptions for this paper is that the CIM and MultiSpeak interfaces will utilize a service-oriented architecture (SOA) and be expressed as a set of WSDL's and XSDs. The WSDLs represent the message "contract" between the sending and receiving system, while the XSDs represent the message payload.

Tools

The tool used for modeling the classes, attributes, and relationships between classes is Sparx Enterprise Architect. Both CIM and MultiSpeak models can be worked with in Enterprise Architect by utilizing the export/import feature of the tool. Enterprise Architect also provides the ability to export models as UML 2.0 compliant XMI for use in other UML 2.0 compliant tools.

A freely available add-in to Sparx Enterprise Architect, CIM EA (cimea.org) was used to develop individual CIM profiles and create the artifacts such as XSDs and RDF files based on the profiles that were created.

Finally, RDF files that have been generated by the CIM EA add-in to Sparx Enterprise Architect are edited and validated in Altova SemanticWorks [5] before being imported into Protégé [6], a web ontology language tool available from Stanford University.

STARTING AT THE TOP

This paper will not cover mapping at the higher level message naming pattern which is covered in more detail here [9], but will focus more on the practical work required to map data at the message payload level.

Both CIM and MultiSpeak use a convention where all classes inherit from a top-level class which gives every subordinate class an identity attribute, among other things. Since these classes are represented throughout any message payload that is developed, any interoperability assessment needs to begin with these top level classes.

The highest level class for the CIM is IdentifiedObject, while the highest level class for MultiSpeak is mspObject. Both classes have an identity attribute shown here in Figure 3, with MultiSpeak using objectID and CIM using mRID. The intent for both of these attributes is that they be unique, so the objectID and CIM mRID easily have a one-to-one mapping. One challenge that should be noted with objectID is that it may not necessarily be globally unique. For example, "a cooperative that hosts an MDM for multiple distribution members could have meters with objectID=12345 for every participant in the system" [10]. So while an objectID may not be globally unique, it is expected to be *contextually* unique, that is, unique within the context in which it is used. Nominally mRID could also have this problem unless a GUID is specifically used for the mRID.

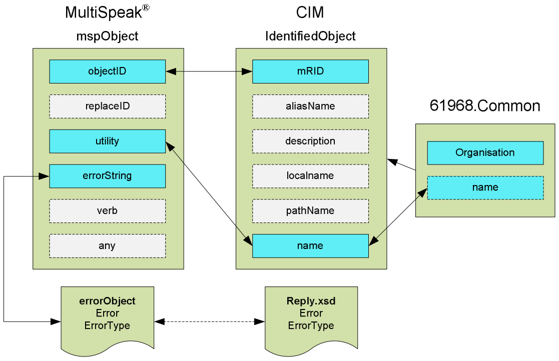


Figure 3 : Mapping between the highest level class of CIM and MultiSpeak

Attribute Mapping Challenges

Attributes that are not mapped to CIM from the MultiSpeak side are replaceID, Verb, and Any. The replaceID is used when the client system may "replace one instance of a data object with another" [3]. There are use cases in prior versions of MultiSpeak where this attribute was used to update the objectID that had been previously used, e.g. replacing a meter, however, it is more appropriate in the current version of MultiSpeak to use a service operation to do an update, rather than rely on this internal attribute.

The Verb attribute is an enumeration indicating client actions; New, Change, Delete, Replace, Link, and Unlink. There isn't a correlation for this type of operation in CIM as the "action" in IEC61968-1-2 implementation guide [7] uses the web service and operation naming convention to indicate action through the use of verbs, i.e., Get, Create, Execute. MultiSpeak also uses verbs in its operation naming convention and the use of the Verb attribute in version 4.1 is to support background compatibility with earlier version of MultiSpeak. Given the CIM to MultiSpeak mapping context this attribute is out of scope.

The use of Any for the purpose of CIM to MultiSpeak mapping must also be considered as out of scope. The challenge with the Any attribute in the context of CIM to MultiSpeak mapping is that it is used to append new attributes to an object that inherits from the mspObject class in MultiSpeak. This convention is allowed in MultiSpeak so that extensions can be made to the MultiSpeak standard if an interface specification does not exist. However, there is no correlation for this type of extensibility in the CIM. As an international standard the CIM follows the guidelines of UN/CEFACT and while this convention, (use of Any) is not expressly mentioned, its preclusion can be inferred by definitions of allowed data types and relationship definitions [11][12]. For interoperability the sender and receiver need to agree on the message payload and this convention essentially is only one step removed from a customized interface, something which should be avoided for obvious reasons. For the purposes of CIM to MultiSpeak interoperability the use of Any is therefore considered *informative* and out of scope.

Two additional MultiSpeak elements that represent a bit of challenge in terms of mapping to CIM are the errorString and the utility attribute. In MultiSpeak the utility attribute is used to denote to which utility an object belongs. The CIM's IdentifiedObject class does not track this information specifically. However, there is an Organisation class which inherits from IdentifiedObject which can contain this information. While the use of Organisation would not necessarily be required for CIM integration, if integrating to a MultiSpeak interface that uses the utility attribute the use of Organisation would need to be accounted for.

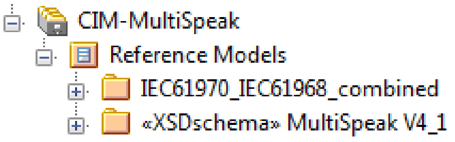
To enable the ability to pass along an error message MultiSpeak uses the errorString attribute. The use of errorString in the highest level class represents a problem of a different sort for CIM integration. This is because rather than using a class to handle this the CIM implementation guidance [7] uses a common Reply.xsd which provides the ability to capture and return any errors generated by the target application. Similarly MultiSpeak uses the errorObject to capture and return errors that may be generated by a MultiSpeak interface [10].

The final challenge for mapping IdentifiedObject and mspObject is the CIM's use of aliasName, description, name, and pathName. These optional CIM attributes represent different ways of using or describing additional information about an mRID or the object that it is indentifying. Since these are optional attributes and there is no specific mapping to an attribute in the MultiSpeak mspObject, these CIM attributes will not be mapped. That being said, the current version of MultiSpeak has added a convention that allows for the persistence of a CIM object by using an identifiedObject class (note the lower case "i" in the naming) that maps to the IdentifiedObject class of CIM.

LOOKING FORWARD: THE CIM AND MULTISPEAK ONTOLOGY PROBLEM

Gruber [4] noted that in computer science that ontology is "an explicit specification of a conceptualisation". Nominally both MultiSpeak and CIM are two representations of a conceptualization whose domain is electric systems. While neither standard is currently expressed in a formalized ontology language such as the web ontology language (OWL), their respective standards represent a rich description of their respective domains. Ideally one would want to bring the CIM and MultiSpeak models into an OWL tool to evaluate the relationships and using a *reasoner*, determine what the inferred relationships or classes may be in an effort to establish where there is commonality between the models and conversely, where significant gaps exist. There are a few challenges with this approach. First both models are very robust so to do an import of the full models and complete analysis would be impractical without first developing a proof of concept. A promising first attempt is an analysis of the MeterReading portions of the CIM and the meterReading hierarchy within MultiSpeak.

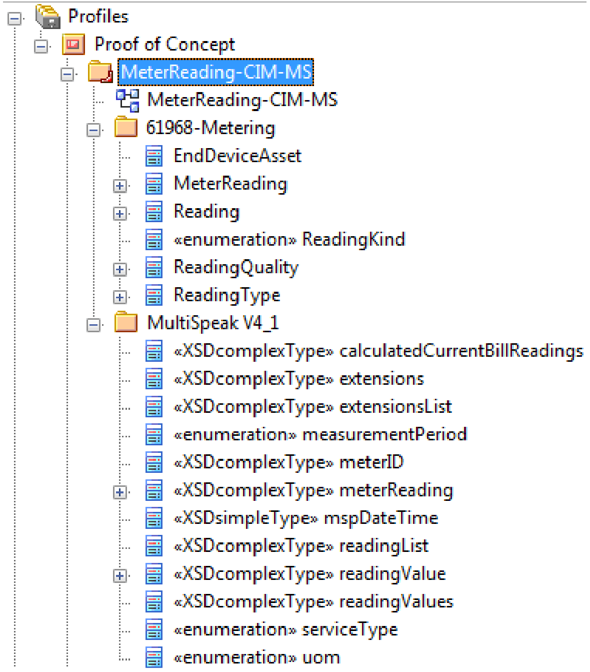
Using the UML 2.0 import feature of Enterprise Architect allows the user to import both the CIM and MultiSpeak standards as reference models.



CIM-MultiSpeak
Reference Models
IEC61970_IEC61968_combined
«XSDschema» MultiSpeak V4_1

Figure 4 : Illustrating the use of CIM and MultiSpeak as reference models within the Enterprise Architect tool

Once each standard is available as a reference model CIMEA can then be used to generate a *profile* which is smaller portion of the larger reference models, allowing the user to refine the area they want to work with as shown in Figure 5.



- Profiles
  - Proof of Concept
    - MeterReading-CIM-MS
      - MeterReading-CIM-MS
      - 61968-Metering
        - EndDeviceAsset
        - MeterReading
        - Reading
        - «enumeration» ReadingKind
        - ReadingQuality
        - ReadingType
      - MultiSpeak V4_1
        - «XSDcomplexType» calculatedCurrentBillReadings
        - «XSDcomplexType» extensions
        - «XSDcomplexType» extensionsList
        - «enumeration» measurementPeriod
        - «XSDcomplexType» meterID
        - «XSDcomplexType» meterReading
        - «XSDsimpleType» mspDateTime
        - «XSDcomplexType» readingList
        - «XSDcomplexType» readingValue
        - «XSDcomplexType» readingValues
        - «enumeration» serviceType
        - «enumeration» uom

Figure 5 : The Proof-of-Concept MeterReading-CIM-MS profile and the reference model packages used to build the profile

Using the CIMEA Add-In a CIM RDF artifact can be generated that creates the RDF subject-predicate-object "triples" in a format that can then be imported into an OWL tool for analysis.

The challenge seen to date is that the CIM RDF, when brought into an OWL tool like Protégé is that the hierarchical relationships, i.e. the associations and inheritance between classes, seen in the reference models, are not preserved when imported in the OWL tool. Further manipulation needs to be done to reestablish this hierarchy. It has not been determined if this is a challenge with Protégé or the structure of the RDF files that are being generated. Once the class hierarchy is reestablished it may be possible to experiment with setting CIM MeterReading equivalent to MultiSpeak meterReading and see what classes and relationships the reasoner may infer.

FURTHER RESEARCH

The CIM first edition was used for this effort and as of this writing the second edition of CIM is being prepared for release with a new round of interoperability testing. Additionally, MultiSpeak v4.1 was used and work continues on that front as well. Mapping from one version to another as was done in this example will only get the industry so far. The industry would be better served for the two standards to merge if possible. Although they both serve the utility domain, they also primarily serve two different constituencies with differing needs.

One solution that has been suggested is that MultiSpeak become a "profile" within the IEC 61968/61970 standard. One way to make this work is for the highest level class in MultiSpeak inherit from the highest level class in CIM. With this solution any existing class in the profile would have the mspObject attributes available to it, and if any CIM classes were utilized to address any gaps in MultiSpeak, the required IdentifiedObject attributes would be available to those classes. A better solution would be to rationalize the highest level class of both standards into a single class, harmonizing any low hanging fruit from the OWL analysis and then create a long-term roadmap for resolving where more significant gaps exist between the standards.

MISCELLANEOUS

REFERENCES

[1] D. Haynes, 2010, "A Case for Optimized Protocols in the Creation of a Smarter Grid", *IEEE Transactions on Power Delivery*, In Press.

[2] G. A. McNaughton, G. Robinson, and G. R. Gray, 2008, *MultiSpeak® and IEC 61968 CIM: Moving Towards Interoperability*, Grid-Interop Forum

[3] G. McNaughton, 2007, *MultiSpeak Version 3.0 Specification,* National Rural Electric Cooperative Association, Arlington, VA.

[4] T. R. Gruber, 1993, *A translation approach to portable ontology specifications*, *Knowledge Acquisition*, vol. 5, p. 199 – 220.

[5] Altova SemanticWorks, http://www.altova.com/semanticworks.html

[6] Protege; http://protege.stanford.edu/

[7] IEC 61968-1-2 Web Services Implementation Profile, International Electrotechnical Commission, [*in press]*.

[8] IEC 61968-9, System Interfaces For Distribution Management – Part 9: Interface Standard for Meter Reading and Control, International Electroctechnical Commission, Geneva, Switzerland.

[9] G. A. McNaughton, 2009, *A Proposed Service Mapping between the MultiSpeak® Specification and IEC 61968-9*, Gridwise Architecture Council, Grid-Interop 2009 Proceedings

[10] MultiSpeak Initiative Technical Committee Meeting Minutes, October 12-14, 2010, *V. Potential for Improvements to Error Development and Processing*

[11] UN/CEFACT, 2009, *Core Components Technical Specification V3.0*

[12] UN/CEFACT, 2009, *Core Components Data Type Catalogue V3.0*