21, rue d'Artois, F-75008 PARIS
http : //www.cigre.org

CIGRE US National Committee
2022 Grid of the Future Symposium

RESTful Microservice Architecture for Electric Utility Distribution Systems

W. MULLENMASTER **T. NIELSEN**
GridBright
USA

M. HIGHFILL **S. STERNFELD**
The Outage Data Initiative
USA

Z. CANDERS
DataCapable
USA

SUMMARY

The most common technologies used by utilities for Operational Technology (OT) system integration technology use information models that are based upon information exchanges that use the eXtensible Markup Language (XML). This technology is less applicable for newly developed systems and applications that use cloud service technologies including Software as a Service (SaaS), and Platform as a Service (PaaS) solutions.

To accelerate the adoption of these modern technologies, improvements for standards such as the Common Information Model (CIM), and MultiSpeak cooperative models. Integration standards, such as CIM and MultiSpeak can evolve to define new RESTful Microservices Architecture (RMA). RMA services created from existing utility standards such as CIM and MultiSpeak, which will help in the adoption of standard RMA technologies. An RMA is comparable to a Service Orientated Architecture (SOA) defined service, except that a microservice differs in the implementation of a service that does less work as simple service with fewer operations. An RMA implementation approach by software solution providers is preferrable, and widespread practice in distributed software modules that are part of distributed computing systems. Microservices in cloud computing used to perform processing for big data and machine learning applications, which improves performance of these systems. Today, there are no defined utility industry standards for RMA application architectures. However, the authors propose why it is necessary and how it is possible to define and implement RMA application architecture standard suitable for the utility industry. This new RMA approach when adopted by standard committees and vendors will create new common models and a new common application microservice architectures in a RESTful way.

KEYWORDS

RESTful Microservices Architecture, CIM, MultiSpeak, Outage Data, ADMS, Emergency Preparedness, Incident Management, DER, DERMS, Interoperability Standard Tools

bill.mullenmaster@gridbright.com

1. INTRODUCTION

Utilities are using and plan to use Distributed Energy Resource Management Systems (DERMS), Distributed Energy Resources (DER)s, and other technologies to support changes in vision for electrification including climate change mitigation and sustainability goals. This requires a heightened role of system and application interoperability to support data exchanges between multiple stakeholders including utilities, generators, grid operators, regulators, and government entities. The emergence of a truly interconnected and a digital grid is clear around the world. Examples in the industry today include:

1. DERMS electric utility company solutions provided by market participants provide consumers with clean interconnected DER and integration with DERMS. These solutions are being used currently in power market systems, including California Independent System Operator (CAISO) [1] and Midwest Independent System Operator (MISO) [2]. Newly created programs and adoption of these programs will introduce new standards into the utility industry that can extend and improve standards that lack information in the U.S. (United States), so that markets can have participation at a distribution level managed with market participants, such as utilities, aggregators, commercial and residential generation. This participation will help make for a more robust, dependable, and sustainable grid, which will require newly developed software and architectures to support them.

2. Ireland's combination of electric utility companies to create Ireland's interconnection with France, as well as integration into the UK National Grid [3]. These new markets are powered in Ireland's interconnection of DER and DERMS systems for photo voltaic (PV), wind, and electric vehicle (EV) charging stations on the island.

As an example of a standard integration between utility OT systems, the CIM Model was used to define outage and incident information for the message layers in as an XML payload for a West Coast electric utility [4]. This standard in integration between the utility's Interactive Voice Response (IVR) and Outage Management System (OMS). The integration implemented as a Simple Object Access Protocol (SOAP) service common in a Service Oriented Architecture (SOA) and not using RESTful services that would be more suitable for modern cloud architectures.

A Service Orientated Architecture (SOA) based upon the Simple Object Access Protocol (SOAP) [5] service is different from a Representational State Transfer (REST) architecture [6]. A REST service Web Application Description Language (WADL) definition is not defined in a Web Services Description Language (WSDL) service contract [7] [8]. REST information can be easily cached due to it being a data driven model that can be used to cache Hyper Text Markup Language (HTML), XML, or JavaScript Object Notation (JSON) [9]. While HTML and XML can be cached using a REST service, the payload context is much larger that HTML or XML. The end tags of HTML and XML doubles the size for the element or object references in the payload content. For example, an objected called **foo** would be represented as ***<foo>who</foo>*** in both HTML and XML, while this would be represented as ***"foo":"who"*** in JSON. The extra end tag needed to complete the object definition in HTML and XML doubles the payload size compared to JSON where the end tag is not used. It is more common for RESTful APIs to be preferred over SOAP in software developed for a cloud native environment [10]. This is due to the ease of use achieved with REST service implementations.

Also, new cloud service technologies platforms make available RESTful API definitions for bucket storage [11]. As a real-world example authors of this paper used bucket storage to implement the outage map integration with an Outage Management System (OMS) for a small utility in the Southeast US [12].

Existing standard models can be adapted to define software industry standard information models that use technologies such as (JSON), which will help create a JSON version of RESTful microservices in the electric power industry. These extended utility industry integration standards can be used in OT systems such as DERMS and directly with DER. Other OT systems where the new integration technology would benefit include energy management systems, market management systems, and emerging incident management platforms for utilities. Utility eco systems of systems can also adopt similar standards for application and integration purposes.

CIM only defines the model, however, for CIM to become more widely adopted, it would be beneficial to define how to implement CIM as standard services. As an implementer of CIM it would be much better to have services defined in a standard, which will make the standard easier to implement. As a software developer, MultiSpeak is often preferrable to CIM due to the ease of implementation of MultiSpeak services. If CIM defined standard services to implement, it would become more widely accepted in the utility industry.

The flexibility of the CIM and MultiSpeak standards do allow for these standards to be extended easily due to their model definitions. However, new tools and technology are needed to help vendors extend these standards and create more lightweight standards, to create information models, such as JSON that can be implemented as RESTful services.

We are proposing what can be done with these power industry standards to help reduce payloads and improve integrations for Real Time (R/T) and Near Real Time (NRT) integrations for power companies. The extensible nature and capabilities of IEC CIM & MultiSpeak are well defined. This extensibility allows for these standards to be extended to incorporate technology standards such as Geographic JavaScript Object Notation (GeoJSON) [13]. This would be an Extension into CIM and MultiSpeak as a standard payload to a JSON formatted IEC CIM and MultiSpeak standard. This is preferrable as a header or attachment to the IEC CIM or MultiSpeak standard. Specifically, this approach provides more flexibility and ease of adoption for vendors that may not have experience with CIM & MultiSpeak in their current form (XML).

To adopt CIM and MultiSpeak REST models and services it will be needed to create new plugins built in Java and/or Java Script to be able to create industry standard models and services that are REST based. These plugins can be used by International and U.S. operated companies, standards advisory boards and committees, regulated and de-regulated power companies, ISOs, software companies, and solution providers.

It would be ideal to have these industry standards tools as plugins available on the open market for technology providers in the power industry to use and implement interoperable standard information models and service architectures for software that is compliant and interoperable. This should also have the ability for testing and certification for companies that choose to implement industry standard models and REST services.

Although the adoption and promotion of REST and JSON services are focused on the first goals to be implemented in industry standard models and service definitions there is little traction for

new innovative technologies that support REST and JSON. By adopting service-oriented architecture, both MultiSpeak and CIM address the interoperability demands of enterprise applications needed to achieve power industry interoperability goals. Both standards enable more prompt and historically better software quality and interoperability. Since they are standard models, they have tools for software development vendors to implement the standard. CIM and MultiSpeak standards and associated involved parties are looking to achieve seamless interoperability of power industry systems and solutions.

CIM covers transmission, generation, and distribution, while MultiSpeak is distribution focused. Furthermore, MultiSpeak originally looked to meet the needs of electric cooperatives in the U.S., while CIM was more focused towards all utilities in the international marketplace. As time progressed, both standards have evolved to serve a broader share of the power industry market for correct usage of power industry data and minimize the need for custom maintenance interfaces.

The CIM (Common Information Model) and MultiSpeak cooperative model standards defined by the IEC and MultiSpeak organizations implemented by technology companies, government regulated and de-regulated power industry entities and companies. It is important to have and improve on the following segments in the power industry with these new standards:

- Standards boards and committees
- ISOs
- ISO market participants
- Power companies
- Software vendors
- Quality Assurance (QA) certification companies
- Consultants

Standards when adopted and certified help interoperability for power industry solutions regulated and non-regulated power prosumers and consumers

- Government and Public Utility Commission (PUC) regulated markets and de-regulated power providers
- Certified software products from technology vendors
- Certified and un-certified integrated systems by solution implementers and utility managed systems

2. IMPROVEMENTS TO UTILITY INDUSTRY STANDARDS

As part of improvements to utility industry standards the process for improvement is defined in Figure 1. The process would be to start with and existing standards, such as CIM or MultiSpeak, and enhance the standard to include more information and generate the standard in a new format such as JSON. That can be achieved by reverse engineering the XML Schema Definition (XSD) into a JSON defined specification for the existing CIM and MultiSpeak standards which are defined in WSDL, XSD, and CIM profiles as an object model. Unlike MultiSpeak, CIM must then be converted to a message format using a tool such as CIM Tool. Due to the complexity of using a CIM profile, creating the XSD's and WSDLs that define a CIM message are not a standard for application development. This can be improved by creating standard services which can be extended to support standard CIM payloads for a new integration architecture.

Improving the tools and framework would include introducing the adoption of REST and JSON services with new innovative plugins to be able to generate the artifacts needed to create REST and JSON models defined in a WADL. When tools become improved and readily available the new standard can be implemented in a Software Development Life Cycle (SDLC) or Agile method.

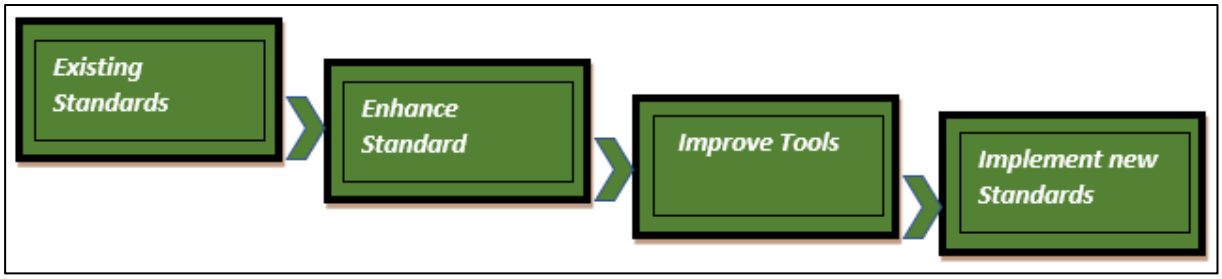

Figure 1. Industry Standard Improvement Model

2.1 Existing Standards and Enhanced Standards

REST Services are defined in a WADL. A WADL is a service contract for REST services, as well as the data model that defines them. One of the goals for improvement of utility industry standards and tools would be to generate REST services as a WADL that can easily be used to produce application software, integration services, and data models in IEC CIM and MultiSpeak.

2.1.1 CIM AS JSON

The benefits of CIM as JSON is that the message can be created as a light-weight model for integration and software development. Migrating an IEC CIM XML message which can be defined as a JSON data model would be achieved by transforming the IEC CIM XSD from a CIM Profile to a JSON definition.

An example Work Cancel Request Message (Payload) existing standard in XML is shown in Figure 2.

```
<act:payload>
        <act:genericCancelRequestPayload>
        <act:wmsWorkId>
          <act:id>1000</act:id>
        </act:wmsWorkId>
      </act:genericCancelRequestPayload>
</act:payload>
```

Figure 2 Example CIM XML message

The response to the Work Cancel Request Message (Payload) is shown in Figure 3.

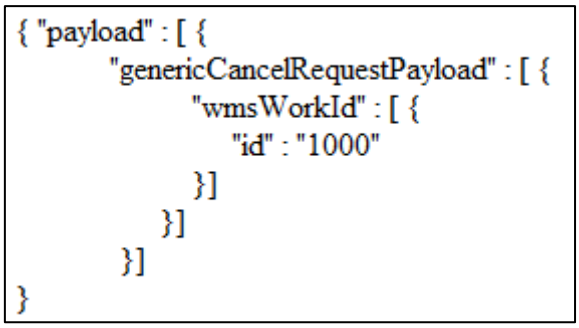

Figure 3 Example CIM XML message

IEC CIM XML message enhanced to a well-defined JSON model is depicted in Figure 3. A REST WADL definition can be created from the converted model in Figure 3. Creating standard REST Services with JSON IEC CIM model payload can create new standard services for IEC CIM. (At this time standard IEC CIM REST services currently are not defined). To be generic the service would accept the "payload" IEC CIM object as a container which can include any IEC CIM message content such as the genericCancelRequestPayload. Creating a set of IEC CIM standard software plugins, the generic payload service can be extended to support multiple IEC CIM messages through inheritance and reflection Software development techniques. This would help make software development easier for creating and defining IEC CIM standard Microservices implemented as a REST service.

2.1.2 MULTISPEAK AS JSON

Benefits of moving from a MultiSpeak XML to a JSON defined message format would also be a more lightweight model. Existing MultiSpeak service definitions that are defined in a WSDL can be transformed into a WADL definition, as MultiSpeak already has well defined services for the models and services the standard defines.

An example, GetActiveOutages, the existing XML standard request message is defined in Figure 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ver="http://www.multispeak.org/Version_4.1_Release">
   <soapenv:Header>
      <ver:MultiSpeakMsgHeader MajorVersion="4" MinorVersion="1"
           MessageID="1" TimeStamp="2022-08-11T14:52:02.000-06:00"
           Context="Emergency Management"/>
   </soapenv:Header>
   <soapenv:Body>
      <ver:GetActiveOutages>
      </ver:GetActiveOutages>
   </soapenv:Body>
</soap:Envelope>
```

Figure 4 Example MultiSpeak XML request

The response to the example GetActiveOutages is shown in Figure 5.

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <env:Header>
        <work:WorkContext
              xmlns:work="http://gridbright.com/soap/workarea/"></work:WorkContext>
    </env:Header>
    <S:Body>
        <ns0:GetActiveOutagesResponse
             xmlns:ns0="http://www.multispeak.org/Version_4.1_Release"
             xmlns:ns2="gml_V4.1_Release"
             xmlns:ns3="http://www.w3.org/1999/xlink" xmlns:ns1="cpsm_V4.1_Release"/>
    </S:Body>
</S:Envelope>
```

Figure 5 Example MultiSpeak XML response message

An enhanced MultiSpeak request message standard to JSON shown in Figure 6 would be defined as the JSON Request which would be a GET HTTP request and be defined as getActiveOutages in the request Uniform Resource Locator (URL).

```
https://mullenmaster-host/GridBrightOutageService/service/getActiveOutages
GET /GridBrightOutageService/service/activeOutages HTTP/1.1
User-Agent: PostmanRuntime/7.29.0
Accept: */*
Postman-Token: 15819524-05e5-4665-837d-426781b45a4c
Host: mullenmaster-host:443
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

Figure 6 Example MultiSpeak JSON message

Moving services to REST services would be implemented as getActiveOutages with the MultiSpeak JSON formatted message defined in Figure 7's response message GetActiveOutagesResult with a list of active event numbers.

```
HTTP/1.1 200 OK
Date: Wed, 24 Aug 2022 22:27:34 GMT
Content-Length: 13
Content-Type: application/json
Access-Control-Max-Age: 86400
Access-Control-Allow-Headers: origin, content-type, accept, authorization
Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS, HEAD
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: *
{
    "GetActiveOutagesResult": {
        "string": [
            "1001"
        ]
    }
}
```

Figure 7 Example MultiSpeak JSON message Response

2.1.3 ADDING GEOJSON AS A STANDARD AND NOT JUST AN EXTENTSION

Adding GeoJSON as an Extension to GetActiveOutages that use Latitude, Longitude, and Geographic Information System (GIS) Device Attributes as a layer to the Outage currently is not defined. This could be added as an extension to the definition of GetActiveOutages which would allow for more standard geographical spatialized integration messages for Advanced Distribution Management Systems (ADMS) managed events to include GIS attribution detail. Please refer to the GetActiveOutages in Figure 5 for a response that could use GeoJSON as a layer to the existing message after the GetActiveOutages object. The GeoJSON standard is implemented by many software vendors and solutions that can easily be converted to existing MultiSpeak messages to gain more adoption of software companies that are not well known in the power industry. The extension of the CIM payload would use the GeoJSON as an object extended into the payload where the FeatureCollection would be another object of the payload. Refer to the example Figure 3 for the payload CIM object. The example FeatureCollection JSON GeoJSON formatted message is defined in Figure 8 as an outage feature collection that is used in incident management systems.


```
{
    "type": "FeatureCollection",
    "features": [
        {
            "type": "Feature",
            "properties": {
                "crew_contacts": "Bill Mullenmaster",
                "ert_source": "I",
                "utility": "GridBright",
                "cause": "Manual",
                "description": "Device Operation (bill)"
       },
            "geometry": {
                "type": "Point",
                "crs": {
                    "type": "name",
                    "properties": {
                        "COORD_SYS": "WSG"
                    }
                },
                "coordinates": [
                    -74.63895597956922,
                    45.52676556430597
                ]
            },
            "id": "1250"
        }
    ]
}
```


Figure 8 Example GeoJSON event notification caused by device operation

2.2 IMPROVE TOOLS

Software development tools available today such as Web Browsers, Eclipse, Visual Studio (VS) Code, SOAP User Interface (SoapUI), and Postman can be used for integration and software development using WADL definitions for REST defined services. Currently not all software development tools support import and export capabilities of WADL service descriptive files to generate REST and JSON artifacts. These tools can be extended to create defined REST services and JSON models defined in a WADL when converted from existing IEC CIM and MultiSpeak standards into a WADL.

As depicted in Figure 9 there are software development tools that can create a WADL to create REST Service Descriptions, however development tools lack the ability to reverse engineer a WADL into software development artifacts that can auto-generate software.

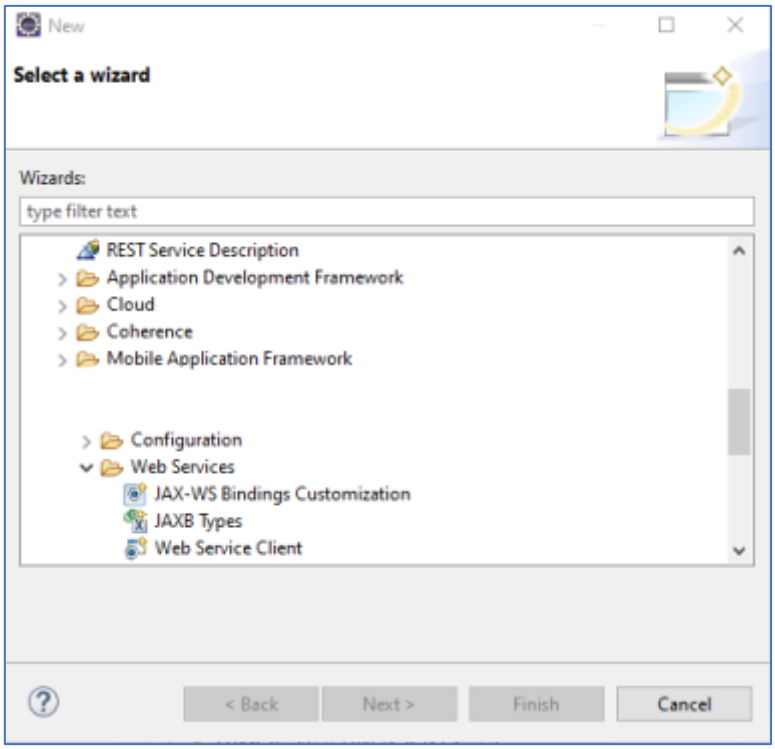

Figure 9 Development tool with REST Service Description (WADL)

Compared to Figure 10 where software development tools support auto generation of WSDL and SOAP services it is easier to create SOA architectures, because there are more available plugins to support them. Software that can be auto generated does help make software development easier and helps companies in the electrical utility industry bring technology to market faster. Autogenerated tools also helps reduce Software defects when code is generated per specifications and standards. If new plugins are created for popular software development applications new standards in REST to support a microservices framework in SaaS and PaaS cloud solutions will be easier to adopt.

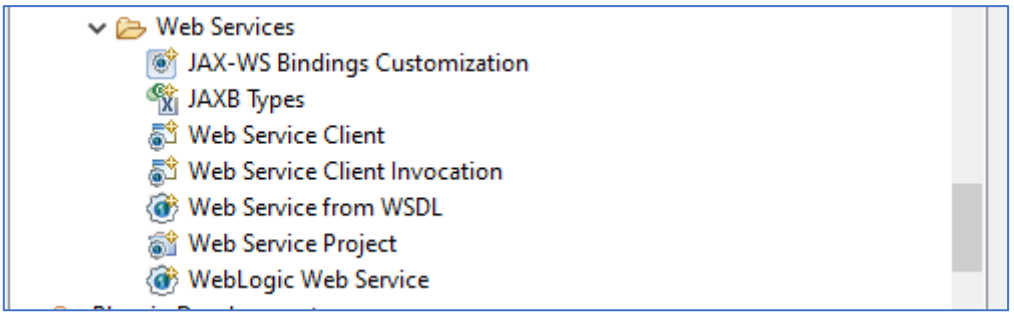

Figure 10 Development tool with WSDL and Web Service generation

2.3 IMPLEMENT NEW STANDARDS

Once the existing CIM and MultiSpeak standards have been converted from XML into JSON and REST services as WADL reference artifacts these new models and services can be implemented in many technology frameworks. For example, in .NET and Java based technologies the WADL can be imported using many software development applications. Once imported, the generated services can be further implemented to create application functionality. In the reference example Figure 11 the Java service implementation was used to create a REST service and implement functionality to support an electric power company's situational awareness information functional requirement for an ADMS system. The artifacts when compiled can be deployed into the proper operating system's application server such as JBoss, or IIS to name a few.

```
/**
 * Retrieves representation of an FeatureCollection GEOJSON which contains layers of Information used for Emergency Response
 * and situational awareness information.
 *
 * @return an instance of FeatureCollection
 */
@GET
@Path("/data")
@Produces({ MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML })
public FeatureCollection featureDataGet() {
    // TODO Auto-generated method stub
    logger.debug("MapRestService::featureDataGet() calling getOutageMapBuilder().build()");

    FeatureCollection featureCollection = getOutageMapBuilder().build(OutageMapBuilder.ALL);
    try {
        ObjectMapper objectMapper = new ObjectMapper();
        String result = objectMapper.writeValueAsString(featureCollection);
        logger.debug("MapRestService::featureDataGet() marshalled FeatureCollection [/n" + result + "/n]");
    } catch (Exception e) {
        logger.error("MapRestService::featureDataGet() exception: " + e.getMessage());
        e.printStackTrace();
    }
    return featureCollection;
}
```

Figure 11. Java RESTful API Code Sample

Compare the Java example in Figure 11 with the .NET code example in Figure 12 for how a RESTful API, can be created to retrieve outage data as a GeoJSON object through a data Microservice [14].

```
Namespace Webservice.REST
{
        [ServiceContract(Namespace = "https://www.geojson.org/")]
        [AspNetCompatibilityRequirements(RequirementsMode =
        AspNetCompatibilityRequirementsMode.Allowed
        //[ FeatureCollectionClass] //Class definition FeatureCollectionClass returns a
        GEOJSON representation as an ASP .NET Service for Information used for Emergency
        Response and Situational Awareness.
        public class FeatureCollectionClass
        {

            public FeatureCollectionClass ()
            {
            }

            [WebInvoke(Method = "GET", RequestFormat = WebMessageFormat.Json,
                       UriTemplate = "/data", ResponseFormat =
                        WebMessageFormat.Json,
                       BodyStyle = WebMessageBodyStyle.Wrapped)]
            public FeatureCollection featureDataGet()
            {
                FeatureCollection featureCollection =
        getOutageMapBuilder.build(OutageMapBuilder.ALL);
                return featureCollection;
            }
            //
            // Continuation of class removed from this example.
            //
        }
}
```

Figure 12 .NET RESTful API Code Sample

3. CREATING JSON TOOLS FOR UTILITY INDUSTRY STANDARDS

Creating JSON tools for working with any utility industry standards includes the definition and creation of supporting services as well. An example reference architecture is shown in Figure 13. The diagram is a reference diagram for both on-premises and cloud hosted solutions that can support RESTful APIs and REST services. When implemented as microservices, PaaS and SaaS system architectures can be defined and created using the proposed RESTful industry standard format based upon CIM or MultiSpeak. These new services can be further distributed in a more detailed design than what is depicted in Figure 13. A more distributed variation would include more services and messaging technologies that would be distributed in the applications. These would support JSON and REST services using technologies like a Java Message Service (JMS) or a Microsoft Message Queueing Service (MSMQs).

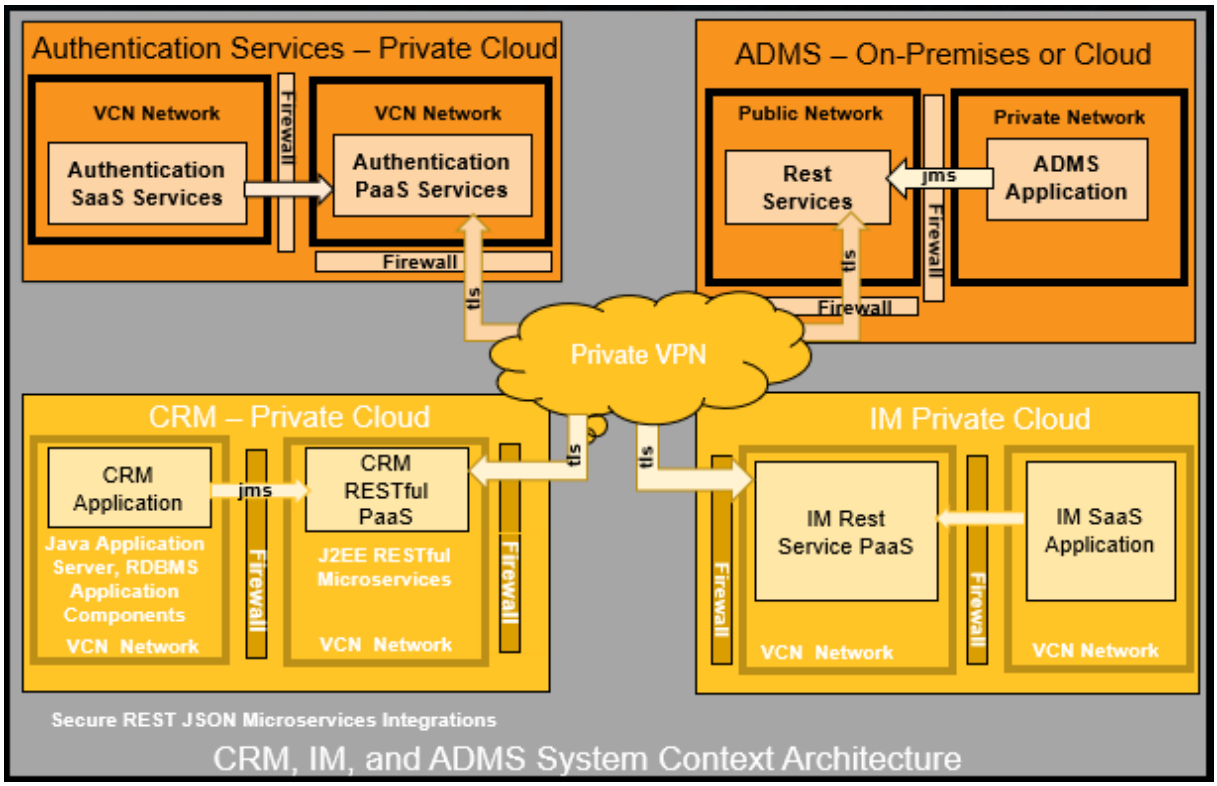

Figure 13 Sample Systems Cloud and On-Premises Architecture

We name a few applications as examples development tools, which are open-source applications and are adopted by many technology providers and software application vendors that serve the power industry. These tools with new application tools or "plugins" could create new REST models and REST services to improve on existing standards which would be implemented to create this new innovative technology. When implemented in a RESTful microservices architecture industry standard created software can be interoperable with both electric utilities hosted, and vendor hosted for power industry cloud hosted applications. For example, as depiction Figure 13, Crew and Resource Management (CRM), Incident Management (IM), and Outage Management Systems are integrated together as a power industry ecosystem can be interoperable when certified against utility standard models like MultiSpeak and CIM, and industry standard GeoJSON models.

Vendors implementing JSON and GeoJSON as a data model for CIM and MultiSpeak based on a proposed standard format can leverage open and widely adopted tools for implementing pre-defined standard models and Microservices in REST.

Once Java or JSP plugins are created, an Integrated Development Environment (IDE) can be used to adopt the plugins. IDEs are available, but the list below has desirable features such as stability, automation features, an effective Graphical User Interface (GUI), an efficient set of tools for debugging and a pre-existing user base:

The following is a list of example Java and JavaScript open-source, free development tools that could implement new plugins for supporting utility power industry Microservices plugins. Code created with these tools can be deployed to native open-source Java Application Servers such as IIS, Jboss, and Apache Tomcat to name a few:

- Eclipse
- NetBeans

- Other open-sourced Java and .NET developer studio tools
- JSP based development tools that support JSP plugins
  - Most common web browsers
  - Microsoft Edge DevTools
  - Community Edition IDEs

These tools can even produce industry standard services that could create a REST model using new plugins to perform conversions to a create new services in a Common Service Architecture (CSA) with a CIM or MultiSpeak JSON Model and REST services. The model and services generated can become even more interoperable for technology providers when also defined by standards committees as a standard REST service names with parameters in JSON for the standard service.

4. INTEROPERABILITY TESTING AND CERTIFICATION

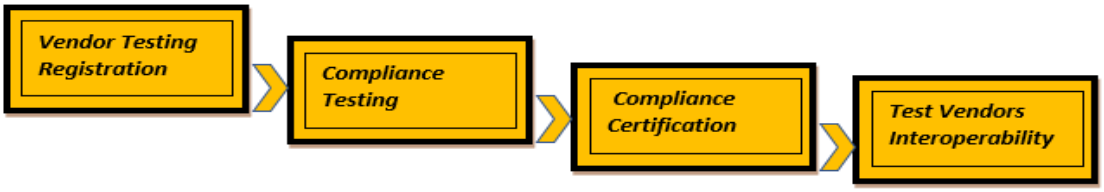

Figure 14 Compliance, Certification & Interoperability Testing

Compliance testing and interoperability testing of new REST technology and JSON defined standards are necessary to certify that all stakeholders are working from the same basis. System implementers from utilities, software companies, consultants, and researchers can help by contributing to interoperability tests. Testing and certification assure:

- "legal" interoperability
- deployed applications meet requirements
- interoperability between vendor applications
- cost control with evergreen application strategy
- inclusion of requirements in procurement (e.g., Request for Proposal – RFP)

Software and integration standards for compatibility and interoperability helps create standards-based software that is not only adopted by technology companies to create new software solutions but makes it easier for utilities to implement integrated solutions that are interoperable. The most efficient path for testing interoperability is first compliance (also known as conformance) testing, then interoperability testing. To be successful, software must have both compliance and interoperability testing.

Compliance testing creates assurance that an implementation complies to a standard or parts of a standard. This testing is an audited environment of accredited test laboratories and certification of results for both positive and negative exchange cases. Conformance testing does not create an assurance that different implementations can exchange additional information.

Interoperability testing evaluates the ability to exchange information. Typically, tests are positive tests and are certified by compliance testers. Interoperability testing does not indicate conformance to the standard [15].

Certification and standards contributions should be published and championed by the power industry, researchers, and vendors that implement the standard. Technology providers and research participation can improve and contribute to new model standards when certified and will help recommend improvements to standard-defined models.

5. PERFORMANCE TESTING RESULTS

When testing the implementations of REST and SOAP services it is apparent that services implanted as a REST service with a lightweight message structure such as JSON does perform better than services that are defined as a SOAP service that utilize XML. As part of the testing performed, three testing scenarios were performed in a controlled software development lab for both REST and SOAP implementations of a MultiSpeak message.

Figure 15 and Figure 16 show the API timings from test executions, which included a manual run of 10 iterations with a second delay between each SOAP and REST service call, and 2 runs of 100 iterations with a 100 milli-second (ms) delay. Figure 14 and supporting detailed information focuses on a SOAP implementation of the MultiSpeak GetActiveOutages service and timings that were found in testing. Figure 16 and information that is documented with it shows the test results for a REST implementation of the MultiSpeak GetActiveOutages for refence to the performance of a SOAP implementation versus a REST implementation of the same MultiSpeak message.

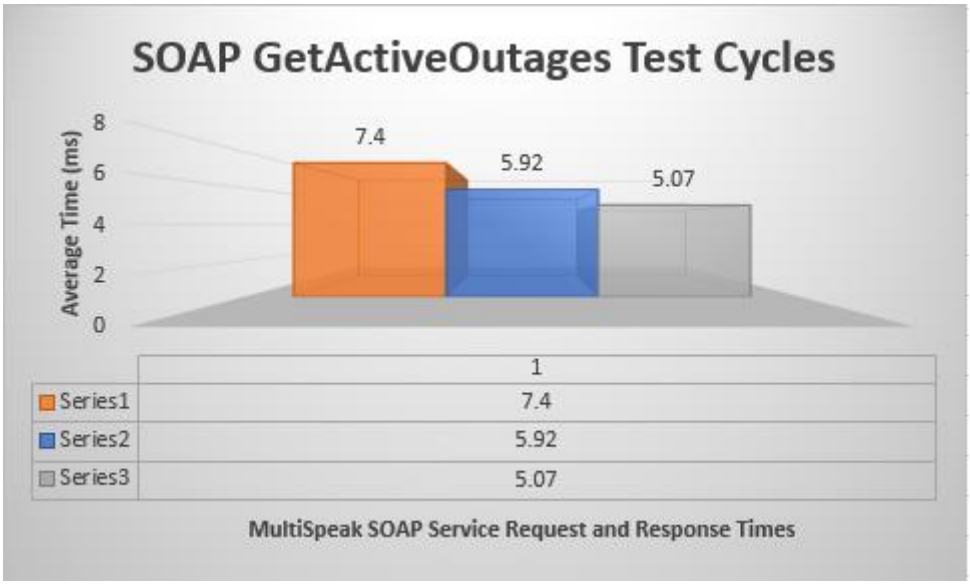

Figure 15 MultiSpeak SOAP GetActiveOutages API timings

The result API timings of the SOAP GetActiveOutages testing:

- Response message payload size: 838 bytes
- Manual Run Test Series 1:
  - Execution count: 10
  - Execution delay between executions: 1 second
  - Total execution time: 74 milliseconds (ms)
  - Average execution time: 7.4 ms
- Automated Run Test Series 2
  - Execution count: 100

    - Execution delay between executions: 100 ms
    - Total time: 592 ms
    - Average: 5.92 ms
- Automated Run Test Series 3
    - Execution count: 100
    - Execution delay between executions: 100 ms
    - Total time: 507 ms
    - Average: 5.07 ms/payload

Figure 15 shows the API timings from test executions for the same GetActiveOutages message using a RESTful architecture.

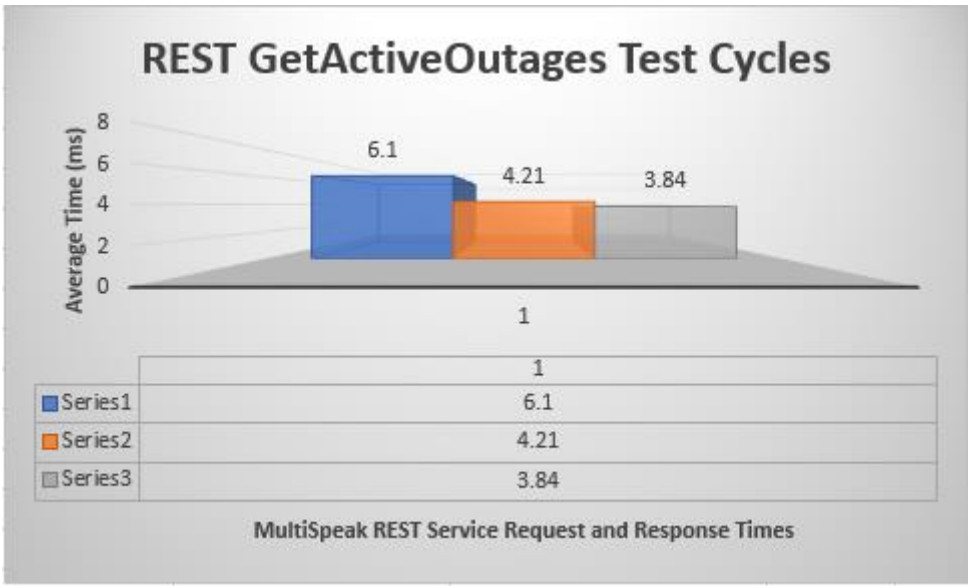

Figure 16 MultiSpeak REST GetActiveOutages API timings

The result API timings of the REST GetActiveOutages testing:

- Response message payload size: 401 bytes
- Manual Run Test Series 1:
    - Execution count: 10
    - Execution delay between executions: 1 second
    - Total execution time: 61 ms
    - Average execution time: 6.1 ms
- Automated Run Test Series 2
    - Execution count: 100
    - Execution delay between executions: 100 ms
    - Total time: 421 ms
    - Average: 4.21 ms
- Automated Run Test Series 3
    - Execution count: 100
    - Execution delay between executions: 100 ms
    - Total time: 384 ms
    - Average: 3.84 ms/payload

As found by these testing results, on average, a small payload REST implemented JSON model services perform faster than SOAP implemented XML data model services. On average this

RESTful API versus the same SOA API for a MultiSpeak GetActiveOutages message performed ~1.38 ms times faster. Whether implementing Real Time (R/T) or Near Real Time (NRT) software REST services outperform SOAP services when using a JSON model for REST service, and an XML model for a SOAP service. This is due to the message size, which depending on the message that a service is supporting could be even faster as a REST JSON service due to the overhead of an XML and SOAP service.

6. DISCUSSION

More research and development for larger models can be found by using larger models in CIM as well as MultiSpeak. Models that support a larger distribution network model such as the IEEE substation and feeder model would be of interest to compare the differences between not only REST and SOAP services, but also comparative to CIM and MultiSpeak models with the same data content. This new research likely will have similar results to the examples that are provide in section 5 PERFORMANCE TESTING RESULTS, however specific timings and performance metrics can be gathered for standard GeoJSON, CIM and MultiSpeak defined models against the IEEE standard distribution model.

More vendors in the power industry's technology sector have already begun to adopt and shift to JSON and REST services. Defining standard models and services with newer technology will create new interoperable systems that can easily be integrated for the power industry. Many grid modernization road maps written by utilities include the adoption of new RESTful Application Programing Interfaces (APIs) [16]. Standards organizations, including the International Electrotechnical Commission Technical Committee 57 (IEC TC 57) [17] and "MultiSpeak Committee Process" [18], need to extend existing standards or develop new standards that are more efficient and natively support cloud integration. The revised and new standards need to able to support new standard application service architecture called RESTful Common Service Architecture (RCSA).

Most software vendors, research organizations, and consultants in the utility industry support and perform compliance testing, compliance certification, and interoperability certification for standard models such as IEC CIM and MultiSpeak. A new standard defined in REST with JSON, if created, will require new testing compliance and certification programs. It would be helpful to review existing programs and continuous improvements to current testing strategies that adopt a more agile framework for compliance and interoperability testing. It will help name technology standards that should follow changes in software development methodologies that are being used by most modern software development organizations. This approach will allow technology solutions to be available sooner to organizations in the Electric Power Industry.

7. CONCLUSIONS

Creating new standards for adoption and creating new tools to support a new standard will help the industry standard and technology solutions by utilizing standards that are being used in commercial software. These recommendations when created will create a new and improved technology that is currently lacking in the power industry, and is innovative to be adopted by software vendors, Information Technology (IT) and business Operational Technology (OT) in the power industry. Leveraging existing testing tools and certification programs will also help achieve these standards and tools for adoption to create new and improved information in a new integration architecture for the power industry. We have proven how a lighter weight standard,

where moving from XML to JSON defined models as a RESTful Common Service Architecture (RCSA) will help to reshape Utility standards such as CIM and MultiSpeak.

BIBLIOGRAPHY

[1] Sidney L. Mannheim, Jeremy Malekos “Docket No. ER06-615-000 Public Version of Annual Report Evaluating Demand Response Participation in the CAISO for 2019”, California Independent System Operator Corporation, January 16, 2020.

[2] Kristin Swenson, “Reliability First, Order 2222 & Order 2222-A”, Midwest Independent System Operator., April 19, 2021. Available: https://rfirst.org/ProgramAreas/ESP/RAPA%20Library/2021-04-19%20MISO%20Distributed%20Energy%20Resources.pdf [Accessed: August 24, 2022]

[3] “Celtic Interconnector: Interconnection project between France and Ireland”, Available: https://www.rte-france.com/en/projects/celtic-interconnector-interconnection-between-france-ireland [Accessed August 24, 2022].

[4] Mullenmaster, W., Nielsen T., Oswald S., DornBrook, A., "OMS (Outage Management Systems) CIM Smart Grid Integration at Seattle City Light.", CIM User Group Meeting, San Francisco, California, October 2010.

[5] Gudgin, M., et. al, “SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)”, World Wide Web Consortium, April 27, 2007. Available: http://www.w3.org/TR/soap12-part1/ [Accessed: August 24, 2022].

[6] Fielding, R. T., “Architectural Styles and the Design of Network-based Software Architectures.”, Doctoral dissertation, University of California, Irvine, 2000.

[7] Sun Microsystems, "Web Application Description Language: W3C Member Submission 31 August 2009". World Wide Web Consortium., August 12, 2012. Available: https://www.w3.org/Submission/wadl/ [Accessed: August 24, 2022].

[8] Booth, D., Liu, C. K., “Web Services Description Language (WSDL) Version 2.0 Part 0: Primer”, The World Wide Web Consortium, June 26, 2007. Available: https://www.w3.org/TR/wsdl20-primer/ [Accessed: August 24, 2022]

[9] Bray, T., “The JavaScript Object Notation (JSON) Data Interchange Format.” Technical Report 8259, IETF, December 2017., Available: http://tools.ietf.org/rfc/rfc8259.txt, [Accessed: August 24, 2022].

[10] Shivpuriya, V., “REST or SOAP in a Cloud Native Environment”, InfoWorld, November 7, 2017.

[11] Amazon Web Services, “What is RESTful API? – RESTful API Beginner’s Guide”, Available: https://aws.amazon.com/what-is/restful-api/ [Accessed: August 24, 2022]

[12] GridBright., “Implementing Oracle NMS at Colquitt EMC”, GridBright. Available: https://www.gridbright.com/project/implementing-oracle-nms-at-colquitt-emc/ [Accessed: August 24, 2022].

[13] Butler, H., et. al, “GeoJSON Specification (RFC7946)”, IETF, August 2016, Available: https://www.rfc-editor.org/rfc/rfc7946 [Accessed: August 24, 2022]

[14] Lianping Chen. "Microservices: Architecting for Continuous Delivery and DevOps"., IEEE International Conference on Software Architecture, Seattle, USA., 2018.

[15] Bas, K., et. al., “The value of CIM conformance testing the basis for interoperability”, UCA International Users Group, September 9, 2020. Available: http://www.ucaiug.org/org/TechnicalO/Testing/CIM_Testing/CIM%20Tests/Webinar-Value%20of%20CIM%20Conformance%20Testing/CIM%20Webinar%209%20Sept%202020%20-%20The%20value%20of%20CIM%20Conformance%20testing%20(1).pdf [Accessed: August 24, 2022]

[16] Siddiqi, U., Dozier, M., Pham, R., Smith, E., "Grid Modernization Plan and Roadmap.", Seattle City Light, April, 2021, Available: https://www.seattle.gov/documents/Departments/CityLight/GridModRoadmap.pdf [Accessed: August 24, 2022].

[17] "IEC Technical Committee Index", Available: http://tc57.iec.ch/index-tc57.html [Accessed: August 24, 2022]

[18] Gary McNaughton, "MultiSpeak® Technical Committee Process", National Rural Electric Cooperative Association, Cooperative Energy Services, October 23, 2012. Available: http://www.multispeak.org/wp-content/uploads/2016/12/Technical_Committee_Process.pdf [Accessed: August 24, 2022]