Pacific Northwest
NATIONAL LABORATORY

PNNL-29393

MS-SPEAK

Final Report and Implementation Roadmap

October 2019

S Pal
CH Miller
TE McDermott
M Engels
SC Tollbom
WJ Hutton

U.S. DEPARTMENT OF ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

PNNL-29393

MS-SPEAK

Final Report and Implementation Roadmap

October 2019

S Pal
CH Miller
TE McDermott
M Engels
SC Tollbom
WJ Hutton

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Summary

This report describes the scope and results of a project undertaken to help consumer-owned electric utilities improve the cybersecurity of their system interfaces. The team developed a cybersecurity reference architecture for the MultiSpeak interoperability standard, which is used by many of these consumer-owned utilities and their supporting vendors. The analysis and use cases focused on the Remote Connect/Disconnect (RCD) interface, and its vulnerabilities. The report suggests five general changes to improve the cybersecurity of MultiSpeak applications:

1. Deprecate some of the specific outdated recommendations of the 2013 MultiSpeak Security Specification, including some of the cited cryptographic methods and transport security layer (TLS) versions.

2. Speed up the adoption of best practices already published by the National Rural Electric Cooperative Association and the National Institute of Standards and Technology.

3. Upgrade to MultiSpeak Version 5, which can support message authentication, subject verification and object verification. Earlier versions cannot.

4. Check, discriminate and arbitrate the use of “universally unique identifiers” in MultiSpeak applications. Despite the name, these are not guaranteed to be unique.

5. Synchronize the clocks between various systems. This would enable more robust message inspection.

An open-source toolkit has also been developed to facilitate customized intrusion detection systems (IDS) for MultiSpeak applications. This report includes a roadmap for others to use and customize this IDS toolkit for additional MultiSpeak versions and interfaces.

Twelve organizations participated in this project’s industry advisory board, including sessions on use of the IDS toolkit and one site visit. Many of the details, including organizational identities and their specific use cases, are proprietary and not included in this public report. Additional project information may be available to users and implementers of MultiSpeak. If interested, please contact the project team or Department of Energy sponsors.

Acknowledgments

This work was funded by the Department of Energy's Cybersecurity for Energy Delivery Systems program. We would like to thank Dr. Carol Hawk, Mr. Ryan Egidi and Jeremy Westbrook of Increase Technologies, Inc., for their support and feedback. We would also like to thank the members of our Industry Advisory Board for their active participation.

Acronyms and Abbreviations

AMI	Advanced Metering Infrastructure
CIM	Common Information Model
CRN	Cooperative Research Network
HTTP	HyperText Transfer Protocol
IAB	Industrial Advisory Board
ICAP	Internet Content Adaptation Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
ISP	Internet Service Provider
MIN	Meter Infrastructure Network
NESCOR	National Electric Sector Cybersecurity Organization
NRECA	National Rural Electric Cooperative Association
RCD	Remote Connect/Disconnect
SOAP	Simple Object Access Protocol
SSH	Secure Socket Shell
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UDN	Utility Distribution Network
UEN	Utility Enterprise Network
UON	Utility Operations Network
VM	Virtual Machine
VPN	Virtual Private Network
WS	Web Service
WSDL	Web Service Definition Language
XML	Extensible Markup Language

Contents

Figures

Tables

1.0 MultiSpeak Security Specification

MultiSpeak® is an interoperability standard used by many consumer-owned electric utilities to achieve plug and play functionality between various products and systems, including smart meters with advanced metering infrastructure (AMI), customer information systems, outage management systems, and others. It has been developed and maintained under the auspices of the National Rural Electric Cooperative Association (NRECA).[1] Beginning with version 3 in 2005, MultiSpeak has enjoyed rapid adoption by rural electric cooperatives and other consumer-owned utilities, with approximately 700 utilities and 40 vendors using versions 3-5 today. Compared to more comprehensive interoperability standards like the Common Information Model (CIM), MultiSpeak has adopted a more pragmatic approach suited to smaller organizations that have constrained information technology staff. Like CIM, MultiSpeak has addressed cybersecurity with an optional add-on called the MultiSpeak Security Specification, released in 2013. That document recommends secure socket layer (SSL) messaging, along with other measures like Web Services Security (WS-Security).

This project was undertaken to help improve MultiSpeak cybersecurity, which would benefit consumer-owned utilities that serve over 25% of the nation's electric meters. It began with a vulnerability assessment of MultiSpeak, focused on a simple but widely used interface that remotely connects and disconnects customer meters. We then developed a cybersecurity reference architecture for MultiSpeak, which shows how and where a purpose-built intrusion detection system (IDS) may be deployed to improve security of the Remote Connect/Disconnect (RCD) functions. An open-source IDS toolkit has been developed so that MultiSpeak users will be able to customize the IDS for other functions. The project has also identified some specific recommendations to lessen MultiSpeak's potential vulnerabilities.

The MultiSpeak protocol is built on top of Simple Object Access Protocol (SOAP), an application communication protocol. In turn, SOAP is built on top of Extensible Markup Language (XML) and Hypertext Transport Protocol (HTTP). The MultiSpeak protocol effectively implements a stateless overlay network between defined endpoints (e.g., an AMI headend). An overlay network is an application-specific virtual network built above an existing network and optimized for a particular application. This creates a problematic cybersecurity environment for maintaining situational awareness. The MultiSpeak protocol is both reliant on and potentially vulnerable to all seven layers of the Open Systems Interconnection Reference Model,[2] from address spoofing through weak authentication to design errors at the top-most layer. It is essential to implement cybersecurity best practices of authentication, authorization, and encryption with MultiSpeak, and to keep all aspects of its cybersecurity up to date. With that in mind, we can make several recommendations to start with:

1. Deprecate some outdated items of the 2013 MultiSpeak Security Specification. Where Transport Layer Security (TLS) or SSL are used, it should only be TLS 1.3 or later. A number of cited cryptographic algorithms, such as 3DES or RC4, are vulnerable to known cyberattacks and should not be used.

[1] http://www.multispeak.org

[2] ISO/IEC 7498-1:1994, "Information technology—Open Systems Interconnection—Basic Reference Model: The Basic Model", Available: https://www.iso.org/standard/20269.html.

2. Reinvigorate the adoption of normative best practices, as already defined by NRECA in 2011.[1] To the best of our knowledge, these standards have yet to be implemented anywhere in the NRECA community. Some best practices are also explained in the National Institute of Standards and Technology Guide to Industrial Control Systems Security.[2]
3. Update to MultiSpeak version 5 with WS-Trust, which provides message authentication, subject verification, and object verification. MultiSpeak versions 3 and 4 do not support sufficient or effective cybersecurity.
4. Furthermore, a MultiSpeak system should be designed to check, discriminate, and arbitrate its use of required universally unique identifiers, which despite the name, are not guaranteed to be unique.
5. Many MultiSpeak users have not synchronized the clocks between their various systems. If not implemented already, time synchronization between all devices should be implemented. This will allow for more robust messaging inspection.

After documenting the normal sequence of RCD messages and responses, we considered a variety of specific risks that must be mitigated. These scenarios expand on some reported by the National Electric Sector Cybersecurity Organization Resource (NESCOR) Technical Working Group 1[3], but customized for MultiSpeak in this project. Table 1 summarizes the risk scenarios. This is not an exhaustive list but includes the scenarios with potentially higher likelihood (since exploits may be relatively easy) or impacts (due to possibility of affecting multiple smart meters). They form a representative list of cybersecurity risks associated with MultiSpeak endpoints.

Table 1: Meter Connect/Disconnect Vulnerabilities in the Triad of Confidentiality, Integrity and Availability

#	Security Risk and Type	Some Possible Impacts
1	Breach of confidentiality	Release of customer information
2	Cyber-physical compromise (Availability)	Revenue loss
3	Credential compromise (Confidentiality, Integrity)	Power outages, including critical loads Revenue loss Extra truck rolls/dispatches Masking criminal activity at customer sites
4	Message replay (Integrity)	Temporary blackouts and cold-load pickups Loss of public confidence in smart metering
5	Man in the middle (Integrity)	Unauthorized reconnections, loss of revenue Extra truck rolls/dispatches
6	Jamming wireless mesh (Availability)	Failure of load control, including demand response Denial of service
7	Request/response flooding (Availability)	Denial of service

[1] NRECA CRN, “Guide to Developing a Cyber Security and Risk Mitigation Plan”, Arlington, VA, 2011, Available at: https://www.smartgrid.gov/files/CyberSecurityGuideforanElectricCooperativeV11-21.pdf.
[2] https://csrc.nist.gov/publications/detail/sp/800-82/rev-2/final
[3] NESCOR, “Electric Sector Failure Scenarios and Impact Analyses – Version 3.0”, Palo Alto, CA, Dec. 2015, Available at: http://smartgrid.epri.com/doc/NESCOR%20Failure%20Scenarios%20v3%2012-11-15.pdf

Figure 1 below shows a notional cybersecurity reference architecture for MultiSpeak, partitioned into four kinds of networks:

1. One or more Utility Enterprise Networks (UEN): UENs provide customer-facing and enterprise services to the utility, such as web sites and email servers.
2. One or more Utility Operations Networks (UON): UONs host important operational systems like customer information, outage management, geographic information, etc. Most of the MultiSpeak endpoints are located here.
3. One or more Meter Infrastructure Networks (MIN): MINs connect smart meters together at the feeder or neighborhood level. MultiSpeak communicates with these devices through an AMI meter headend, which may use proprietary communication protocols to the devices.
4. One or more Utility Distribution Networks (UDNs): UDNs connect the local meter infrastructure networks into the utility enterprise. These are different from the electric power distribution system, unless the utility happens to use power line communication. Fiber, microwave, radio, and internet service providers are also used.

The reference architecture helps to plan security measures that fit the utility's resources and infrastructure choices. There are many options (e.g., cloud hosting, software as a service) for each network and system. In addition to firewalls between each network zone, Figure 1 highlights two specific recommendations for the MultiSpeak RCD function:

1. Virtual private networks (VPNs) should be used for all communication flows between the meter infrastructure networks and utility networks. The vendor-proprietary connections between meters and headends should also be secured with VPN or TLS.
2. IDS with custom business rules should be deployed in the utility operations networks. Unless the utility has an Enterprise Service Bus, and most MultiSpeak users do not, the IDS also needs to be deployed somewhere in the meter infrastructure networks. This may require support from the AMI vendors.

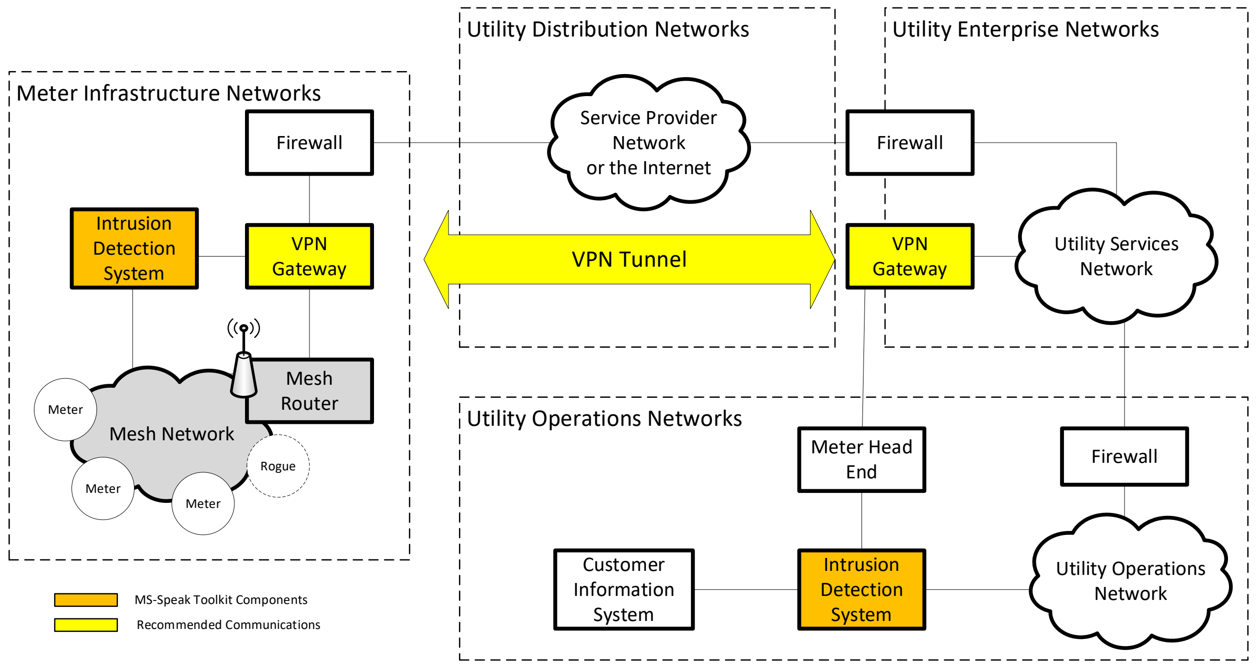

Utility Distribution Networks
Utility Enterprise Networks
Meter Infrastructure Networks
Firewall
Service Provider Network or the Internet
Firewall
Intrusion Detection System
VPN Gateway
VPN Tunnel
VPN Gateway
Utility Services Network
Mesh Router
Meter
Mesh Network
Rogue
Meter
Meter
Utility Operations Networks
Meter Head End
Firewall
Customer Information System
Intrusion Detection System
Utility Operations Network
MS-Speak Toolkit Components
Recommended Communications

Figure 1. Cybersecurity Reference Architecture for MultiSpeak with VPN and Intrusion Detection Systems

In this project, a user-customizable IDS was developed on an open-source foundation so that consumer-owned utilities can employ packet inspection techniques at a reasonable cost to help secure MultiSpeak. The IDS will reliably detect RCD requests that violate any of the rules, create logs, and generate alarms as appropriate. For example, the IDS can cross-check with other systems (e.g., billing), check for correct sequencing of operations, limit the number of operations per time period, ignore cold-weather disconnects, etc. If the utility uses an Enterprise Service Bus, the IDS will forward only the authenticated messages. The IDS could be further generalized to inspect other metering-related messages and other MultiSpeak interfaces.

2.0 MS-SPEAK Toolkit

As part of the MS-SPEAK project Pacific Northwest National Laboratory (PNNL) has developed an open-source specification-based Intrusion Detection System (IDS) for MultiSpeak request messages that violate established business rules. The initial implementation of the IDS has focused on the RCD function to detect RCD connect, reconnect or disconnect packets that violate the utility's policies, which are specified as part of the business rules. In order to enable testing of the IDS by the developers as well as potential users, PNNL has also developed an MS-SPEAK toolkit that consists of three Qt-based components (see Figure 2) as follows:

1. MultiSpeaker: This is the MultiSpeak endpoint, referred to as "Alice".

2. MultiSpeakerServer: The server listening to MultiSpeak messages, referred to as "Bob".

3. IDS or IdsEditor: The IDS is built to secure MultiSpeak interfaces and the business rules are specified by the user using the interface called the "IdsEditor". The IDS is also referred to as "Irene".

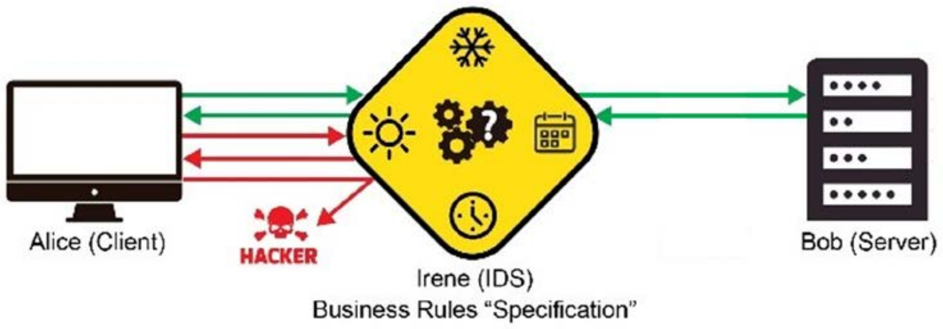


Figure 2. Three Components of the MS-SPEAK Toolkit.

The MS-SPEAK toolkit utilizes two open source software packages:

1. It uses the open source software package squid[1], a full-featured HTTP proxy with caching and forwarding functions. This provides access control, authorization and logging environment to aid in the development of web proxy and content-serving applications. It is typically run in the background as a daemon service and can be used in certain security applications (like Irene IDS) by efficiently filtering content on the fly.

2. The toolkit also uses c-icap[2], which is an open source software package to implement the Internet Content Adaptation Protocol (ICAP) server. ICAP is a lightweight HTTP-like protocol specified in RFC 3507[3], which is used to extend transparent proxy servers (i.e., squid), thereby freeing up resources and standardizing the way in which new features are implemented. c-icap can be used with HTTP proxies that support the ICAP protocol to implement content adaptation and filtering services. Here, content adaptation refers to performing a particular value-added service (e.g., content manipulation) for an

[1] http://www.squid-cache.org/
[2] http://c-icap.sourceforge.net/
[3] https://tools.ietf.org/html/rfc3507

associated client request/response. c-icap typically runs in the background as a daemon service but can be run in foreground for debugging purposes.

The ICAP server is used with the HTTP proxy package squid to realize content adaptation and filtering for MultiSpeaker. In practice, any MultiSpeak package sent with MultiSpeaker is sent to the squid proxy which passes it on to the ICAP server for inspection and filtering based on business rules configured with the IdsEditor. The filtered packets are then sent on to the end point via squid. See Figure 3 for an overview of the message traffic.

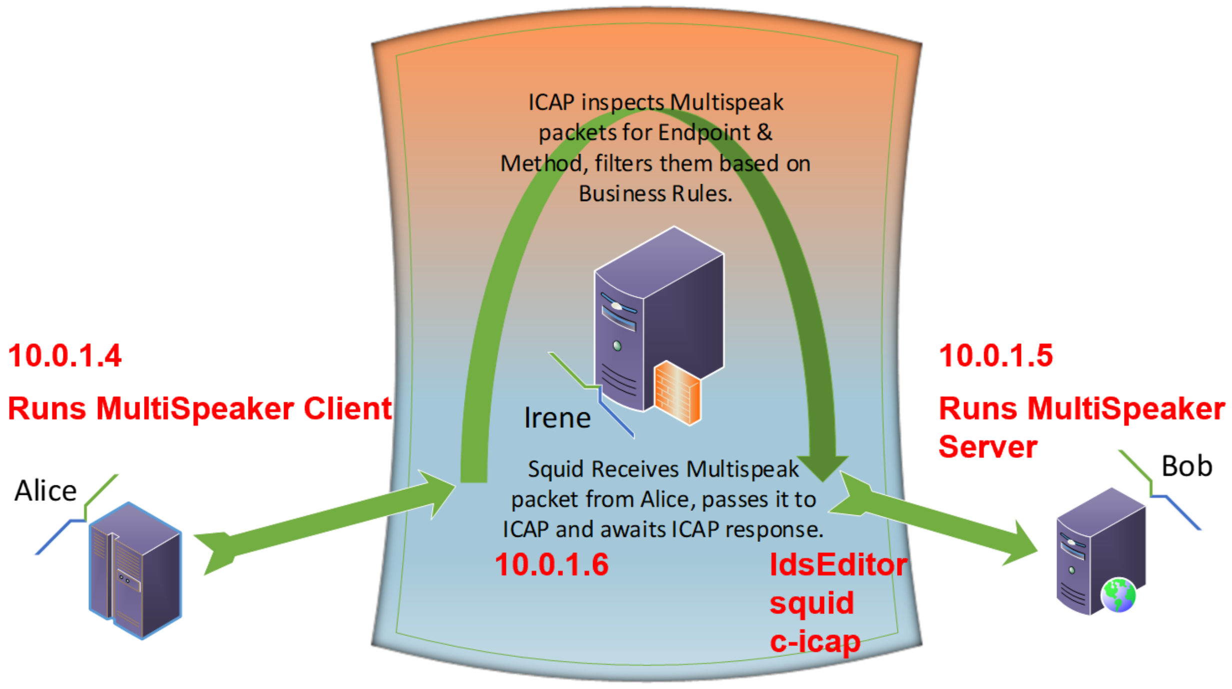

ICAP inspects Multispeak packets for Endpoint & Method, filters them based on Business Rules.
10.0.1.4
Runs MultiSpeaker Client
Alice
Irene
Squid Receives Multispeak packet from Alice, passes it to ICAP and awaits ICAP response.
10.0.1.6
IdsEditor
squid
c-icap
10.0.1.5
Runs MultiSpeaker Server
Bob

Figure 3. Information Flow between MS-SPEAK Components on Three Virtual Machines

2.1 MultiSpeaker[1]

MultiSpeaker is a client application for generating crafted MultiSpeak® request and response messages. It is possible to specify the following parameters, which are detailed in Appendix A:

- Type of endpoint: “Function Blocks” list the different possible endpoints (e.g. MR, or Meter Reading) by functional use.
- Type of method: The type of method can be selected from the “Methods’ menu. For any selected endpoint there will be associated method(s) in this menu.
- Topology: The selected endpoint can be dragged from the “Function Blocks” menu to the “Topology” space to create the desired topology.
- Timeline of MultiSpeak generated events in order of execution. An “event” by default populates a Request/Response pair of communications messages and it is possible to edit the details (e.g. time of request message or delay of response message).
- It enables selection of the appropriate Web Service Definition Language (WSDL) from the “WSDLs” selection option.

[1] https://github.com/pnnl/ms-speak/tree/Phase2/Multispeaker

2.2 MultiSpeakerServer

The MultiSpeakerServer is a server application used to listen for MultiSpeak® requests and return response messages.

2.3 IDS

The “Irene” IDS is an open-source application that is aware of the MultiSpeak protocol and used for filtering MultiSpeak request messages in order ensure that these messages do not violate the business rules. Business rules for the IDS can be added, deleted or edited using the IdsEditor interface. The existing version of the IdsEditor has been developed to serve as a simple example that currently accepts a range of values for specified parameters, such as time of day, temperature, and number of allowed RCD requests per day. For example, a utility may only issue about 5 remote disconnects a day, usually around 10:00 a.m. So, business rules can be set up to accept only 5 messages per day within a specified time window on any day; any additional messages would be rejected. In another example, a utility may not issue a remote disconnect if the ambient temperature is below 35 °F or above 80 °F; therefore, business rules can be set to accept remote disconnect messages only when the temperature is within that specified range.

The business rule changes are written to a configuration file, and it is possible to create new configuration files. If Irene IDS detected otherwise-valid disconnect messages that were outside of this rate or time, the IDS could take user-defined action, including; log the message, forward the message, or drop the message. Irene is not limited to message monitoring activity only, but can execute user-defined actions, much like an Intrusion Prevention System.

Typically, MultiSpeaker would be installed on the same machine as squid and icap. Squid receives MultiSpeak packets from MultiSpeaker, passes it to ICAP and awaits its response, while ICAP inspects MultiSpeak packets for endpoint and method, and filters them based on the specified business rules. If upon inspection ICAP determines that the request message does not satisfy the specified business rules, then Squid is provided a negative ICAP response based on which the request message is not forwarded to MultiSpeakerServer and instead an error status is returned to MultiSpeaker. Squid directly passes MultiSpeak response messages from MultiSpeakerServer to MultiSpeaker, and ICAP is not currently configured to receive response packets from Squid for filtering.

It should be noted that currently the IDS is not performing deep packet inspection. It is parsing the message headers and not looking any further than the method and the endpoint. However, it is possible to further develop the ICAP module with deep packet inspection in order to determine meter ID or IP address. The IdsEditor can be further developed to enable setting up of more complex business rules based on the needs of the utilities. It is also possible to send message logs to specific destinations through customization of the ICAP server.

3.0 Results

In this Section the approach that has been adopted for the different project tasks is summarized and the accomplishments are listed.

3.1 Task 1 – Define Attack Scenarios and Architecture

As part of Task 1, the team analyzed the inner workings of the MultiSpeak protocol, RCD messages and responses. We then implemented a cybersecurity analysis of MultiSpeak Version 5 as it applies to RCD function.

The layered architecture of MultiSpeak tends to increase the attack surface, and without implementing cybersecurity best practices of authentication, authorization, and encryption, the MultiSpeak protocol is inherently vulnerable to numerous common vulnerabilities. Therefore, it is necessary to take a more holistic view of cyber security in MultiSpeak. Potential cyber threats, both current and emerging were determined and assessed. Some of these scenarios expanded on certain scenarios reported by the National Electric Sector Cybersecurity Organization Resource (NESCOR) Technical Working Group 1; we adapted them for MultiSpeak applications and our selected endpoints. Instead of developing an exhaustive list of cyber-risk scenarios, the team focused on those scenarios associated with MultiSpeak RCD function that have potentially higher likelihood (since exploits may be relatively easy) or impacts (due to possibility of affecting multiple smart meters). These cyber-risk scenarios can be leveraged by electric cooperatives to improve cybersecurity awareness during procurement/planning, to conduct risk assessments, and to test the cybersecurity of the RCD system during deployment. These scenarios can also serve as templates when testing the cybersecurity of MultiSpeak end-points other than RCD.

A cyber security reference architecture was also developed with the aim to architect a more secure network that will result in better defense of a MultiSpeak dependent system against existing and emerging cyber-risks. The utility’s network architecture typically includes control stations, sensors, firewalls and gateway systems. The architecture is composed of four types of networks including - UEN, UON, UDN, and MIN. Figure 4 illustrates a notional function-based cyber security network architecture, with more system identification detail than was presented in Figure 1. The best architectural approach for each network type in a given service area will depend on costs, resources, and infrastructure available to the utility.

As part of this task, recommended cybersecurity best practices have also been provided for strengthening the security of the MultiSpeak dependent systems.

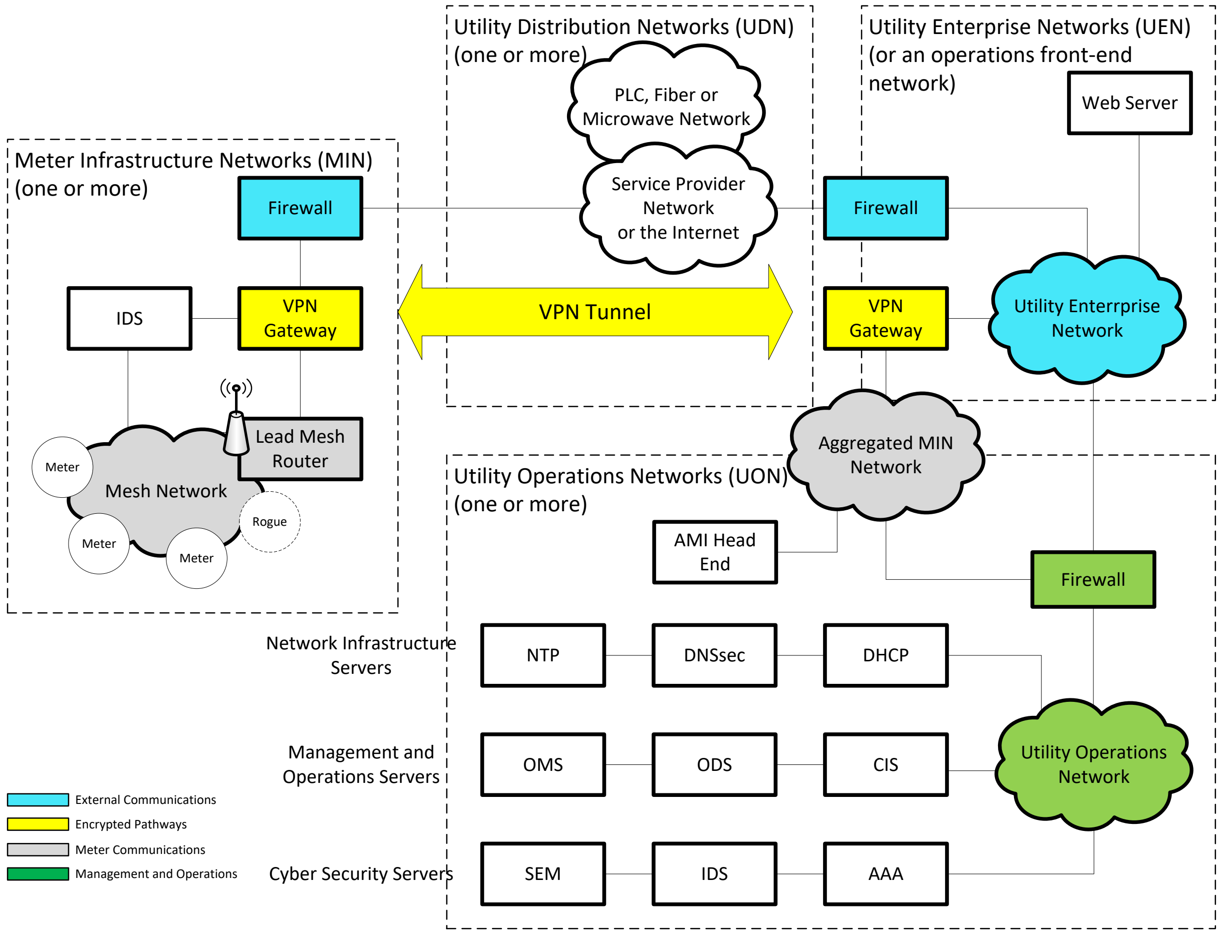

Utility Distribution Networks (UDN) (one or more)
PLC, Fiber or Microwave Network
Service Provider Network or the Internet
Utility Enterprise Networks (UEN) (or an operations front-end network)
Web Server
Meter Infrastructure Networks (MIN) (one or more)
Firewall
Firewall
IDS
VPN Gateway
VPN Tunnel
VPN Gateway
Utility Enterrprise Network
Lead Mesh Router
Meter
Mesh Network
Rogue
Meter
Meter
Aggregated MIN Network
Utility Operations Networks (UON) (one or more)
AMI Head End
Firewall
Network Infrastructure Servers
NTP
DNSsec
DHCP
Management and Operations Servers
OMS
ODS
CIS
Utility Operations Network
External Communications
Encrypted Pathways
Meter Communications
Management and Operations
Cyber Security Servers
SEM
IDS
AAA

Figure 4. Cybersecurity Reference Architecture for MultiSpeak with Utility Subsystem Details

3.2 Task 2 – Develop and Implement a Test Platform

As part of Task 2, a user-customizable IDS has been developed on an open-source foundation so that consumer-owned utilities can employ RCD request inspection at a reasonable cost to help secure MultiSpeak. The IDS will reliably detect RCD requests that violate any of the business rules specified by the user, create logs, and generate alarms as appropriate. For example, the IDS can limit the number of operations per time period, ignore cold-weather or hot-weather disconnects, etc. The IDS could be further generalized to inspect other metering-related messages, and other MultiSpeak interfaces.

Leveraging the MS-SPEAK toolkit, the performance of the IDS was tested by implementing different scenarios. The IDS was able to accurately detect RCD request messages that violated any of the business rules specified in the IdsEditor.

3.3 Task 3 – Vendor Engagement and Support

During the TechAdvantage conference which occurred between March 10-13, 2019, the members of the MS-SPEAK team shared an overview of the project with approximately 75 vendors, gauged their interest, and gathered their initial thoughts and inputs.

In addition to the vendors, some of the WA consumer-owned utilities also expressed interest in collaborating with PNNL to adapt the IDS for their needs and testing it.

The findings from Task 1 and 2 were also presented by our project partner Increase Technologies at the MultiSpeak Technical Committee meeting on April 25th, 2019, located in Glenwood Springs, CO. The toolkit and the performance of the IDS were demonstrated to the meeting participants, and inputs/comments were collected at the end of the presentation. Overall, vendor reception to the presentation was positive and attendees recognized the value of a tool that has visibility to all communications on a network, rather than only visibility to a single point-to-point interface. Based on the feedback received from the members of the MultiSpeak Technical Committee the following major changes were made in the IDS before further demonstration:

- Return messages from the IDS were changed to be properly formatted MultiSpeak response messages. This would mean returning the appropriate response message defined in the MultiSpeak WSDL for the method in question. For example, if an InitiateConnectDisconnect message is sent and rejected the sender should receive a properly formatted InitiateConnectDisconnectResponse message containing an appropriate MultiSpeak error code. For implementing this change MultispeakerServer basically modifies the MultiSpeak message it receives, inserts an error code and applies a MultiSpeak Response Header.
- It was pointed out that the rejection message format contained more information than necessary, and the additional information may be used by an attacker to hone an attack. For example, the response was indicating the use of an ICAP server and also mentioned its version. However, in keeping with cyber-security best practices the error messages should be as minimal as possible, providing only the essential information. Accordingly, the response message was crafted to contain only the MultiSpeak Response Header, with an appropriate replyCodeCategory inserted to indicate the error.

Based on all of the above outreach activities, the MS-SPEAK team recruited interested stakeholders to the project's Industry Advisory Board (IAB). In the months of July and August 2019 bi-weekly web conferences were organized with the IAB members for describing the toolkit, demonstrating the IDS, collecting the requirements of the different vendors, and ironing out the test procedures. The IAB included eleven participating vendors and utilities, plus NRECA and their consultant, Increase Technologies. Four web conferences took place on July 12, July 26, August 9, and August 23, 2019, where several IAB members joined each of these conferences.

The IAB members from various organizations were asked about their requirements (if any) regarding the Operating System, MultiSpeak versions, and endpoints. Based on the responses received, the MS-SPEAK team has continued further updating the toolkit. The toolkit is currently developed to work with MultiSpeak Version 5, Linux OS and the RCD endpoint.

Some of the major takeaways regarding the needs of the IAB members are as follows:

- Removal of Java dependency and replaced by Qt-based components,
- Better logging, handling of encryption, provision for high availability, and integration of more complex business rules, and
- Allow for Windows compatibility or implementation on the cloud.

The first and third items were implemented, and keeping project timeline in mind portions of second item have been implemented. As part of possible future work, second item can be completed which will help further develop the IDS for meeting the requirements for real-world deployment.

3.4 Task 4 – Workshop

A test environment was created on Microsoft Azure, consisting of 3 virtual machines (VM) running Ubuntu 18.04 LTS. This cloud-based test environment was made accessible over the Internet using Secure Socket Shell (SSH), and all participants were provided secure access so that they could perform their own testing. A video was also created that described all the components of the toolkit, how to connect to the test environment and demonstrated a test case. The video was shared with the potential workshop attendees and later uploaded to the Git repository[1]. The example test case helped the participants get acquainted with the IDS and the test environment, and the participants could customize the IDS configurations to adapt it for their own testing. On August 29 a virtual workshop was presented from PNNL.

On September 9, the PNNL MS-SPEAK team visited a consumer-owned utility to discuss a use case involving other specific vendors and interfaces. There is a specific cybersecurity need at this location. Requirements and sample message packets were captured. If implemented, this use case could become the first field application of the MS-SPEAK toolkit, and possibly encourage others to adopt the toolkit.

The interactions during the four web conferences, the virtual workshop, and the site visit, helped the team better understand the needs of potential adopters of this IDS. Based on the responses of the different stakeholders, which served as a list of their requirements, the team created a roadmap that can be achieved with project carry-over funds. This roadmap has been discussed in greater detail in Section 4.

3.5 Task 5 – Documentation and Reporting

This report provides public documentation and reporting of the work that has been done as part of the MS-SPEAK project. In addition, source code and documentation for the MS-SPEAK toolkit have been posted on GitHub.

[1] https://github.com/pnnl/ms-speak/blob/Phase2/Multispeaker/videos/MS-SPEAK-on-Azure.mp4

4.0 Roadmap

The MS-SPEAK project team has completed all the tasks and achieved the objectives of the MS-SPEAK project. The outreach activities that have been performed at the different stages of the project helped further our understanding of the various needs of the potential stakeholders. Based on this list of requirements the team has developed a roadmap that will help better secure MultiSpeak applications.

The items in the roadmap are as follows:

- The MS-SPEAK team has performed a cybersecurity analysis of the MultiSpeak protocol and analyzed a number of potential cyber risk scenarios. This information can help in increasing awareness regarding the cybersecurity issues associated with MultiSpeak application without implementation of security best practices or dedicated IDS. As a first step, stakeholders can contact PNNL to get access to the report to educate themselves on the findings and be more aware of the potential cyber-risks.

- The MS-SPEAK team has developed a list of highly recommended best practices aimed to prevent, detect and mitigate cyber-attacks. Vendors and utilities implementing MultiSpeak should carefully consider and incorporate the relevant best practices. It is also recommended that instead of using the older versions of MultiSpeak, the latest version 5 of MultiSpeak should be implemented wherever possible.

- The MS-SPEAK team has developed an entire suite of tools that we are calling MS-SPEAK toolkit for enabling implementation of different cyber-risk scenarios and testing of the IDS. A cloud-based test environment was later created on Microsoft Azure, and can be made accessible to interested stakeholders over the Internet using SSH. Interested vendors or utilities should contact PNNL to gain secure access to the test environment, customize the toolkit based on their specific business needs, and perform cybersecurity testing of their MultiSpeak applications or the baseline/demonstration IDS that PNNL has developed. The components of the toolkit are available in the GitHub repository.

- As an alternative to using the cloud-based test environment, the stakeholders can build their own test platform on Linux from the scripts located on GitHub, which were the same scripts used to build the test virtual machines on Azure.

- The current implementation of the IDS focuses on MultiSpeak RCD messages and detects any business rule violations in RCD requests. This same script can however be used as a template and adapted to perform detection of business rule violations in any functions other than RCD. The response can be customized to integrate any desired actions.

- Currently, the toolkit is not using accurate time or temperature information. However, for implementing the IDS to a real-world MultiSpeak application, the IdsEditor should be programmed to integrate accurate time and temperature information. The filters specifying the maximum number of acceptable request messages and acceptable time window in a day should be adapted to the needs of the utility.

- At present the IdsEditor has very rudimentary business rules, intended to serve as easy-to-follow examples. Stakeholders can customize the IDS to add, modify or delete business rules.

- Currently if a request message does not satisfy the specified business rules, then the request message is not forwarded to MultiSpeakerServer and instead an error status is returned to MultiSpeaker. The utilities implementing the IDS can customize it to perform message blocking or provide customized alarms to the specified destination.

- If any utility needs to handle large numbers of rules by meter, then provision for meter-number based rules can be added to the IDS. Custom business rules can be built, then relevant configurations can be added into the MS-SPEAK source code, which is written entirely in C++.

- More open-ended customizations of the toolkit by others would follow this pattern:

    - Define the scenarios, i.e. use cases, that will be mitigated. Denial-of-Service and other vulnerabilities should be included along with site-specific interoperability and security concerns. The IDS component and business rule locations (e.g., Figure 4) and the network configuration (e.g., Figure 3) should be defined up front. The actions taken by the IDS (e.g., logging and email notification) should also be decided. These will all impact requirements of the IDS.

    - Identify the MultiSpeak vendors, products and versions involved in the use case. During this process, identify the key messages and payloads that the IDS will have to inspect.

    - Design any new business rules as appropriate, building on the RCD example rules based on time-of-day, weather and rate-of-messaging. More complicated rules could reference the sequencing of messages, for example.

    - Follow an iterative development and testing process:

        - First, the public GitHub repository should be forked to a private repository.

        - Modify the open-source MS-SPEAK components as needed.

        - Deploy the modified MS-SPEAK system onto a network for testing.

        - Define and execute test scenarios using the IDS with MultiSpeak systems.

        - Refine the requirements, code and test scripts as needed.

    - Incorporate the customized MS-SPEAK components and business rules into vendor code or utility site code.

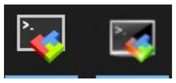
    - Alternatively, the implementer may choose in-house, commercial or open-source libraries different than what MS-SPEAK uses. In this case, the MultiSpeaker suite can still be useful in generating sample messages and responses for test scenarios.

Appendix A – MultiSpeaker and MS-SPEAK Documentation

This appendix summarizes the user interface for components of the MS-SPEAK toolkit. On-line user documentation is maintained on GitHub[1], along with a tutorial video[2]. The software runs on Linux and it can be built from source. It can also be accessed on a cloud-based test platform. For cloud-based access from Windows, an X11 server such as MobaXterm[3] is recommended.

Each MS-SPEAK component is described through a series of screen shots as they appear on Windows. Buttons and other interface features are described with reference to red numerical labels on the figures.

A.1 MultiSpeaker Server



Once this application opens, you can find its X-window icon in the Windows Taskbar, next to the similar MobaXterm icon. The main interface appears in Figure 5.

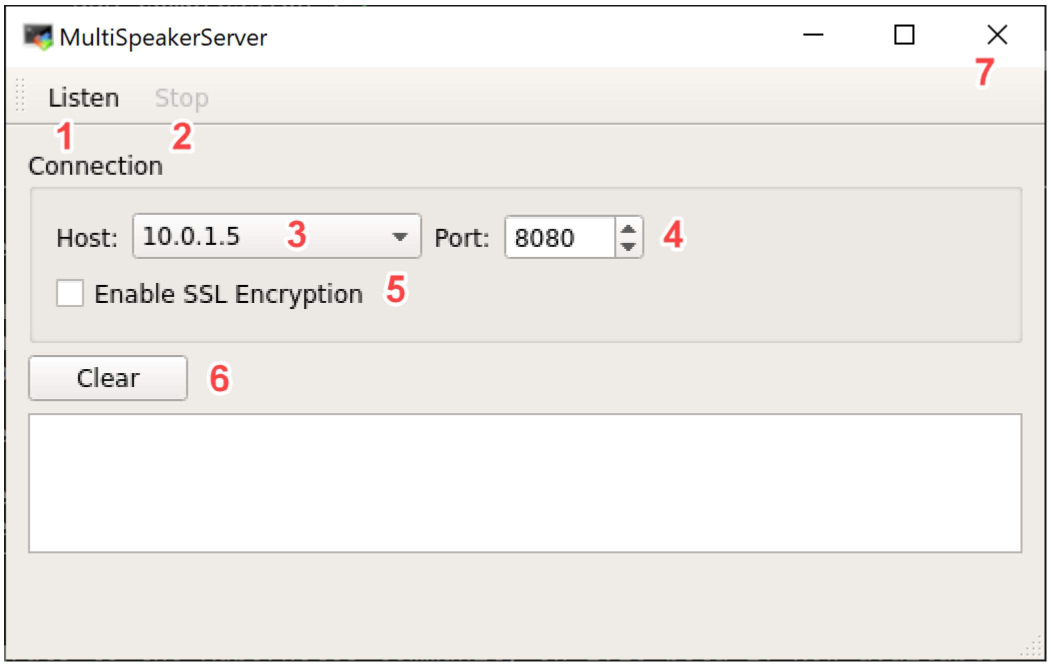


Figure 5. MultiSpeakerServer Interface

1. For activating the application so that it can listen to connections.
2. For stopping the listening mode.
3. For setting the host address.

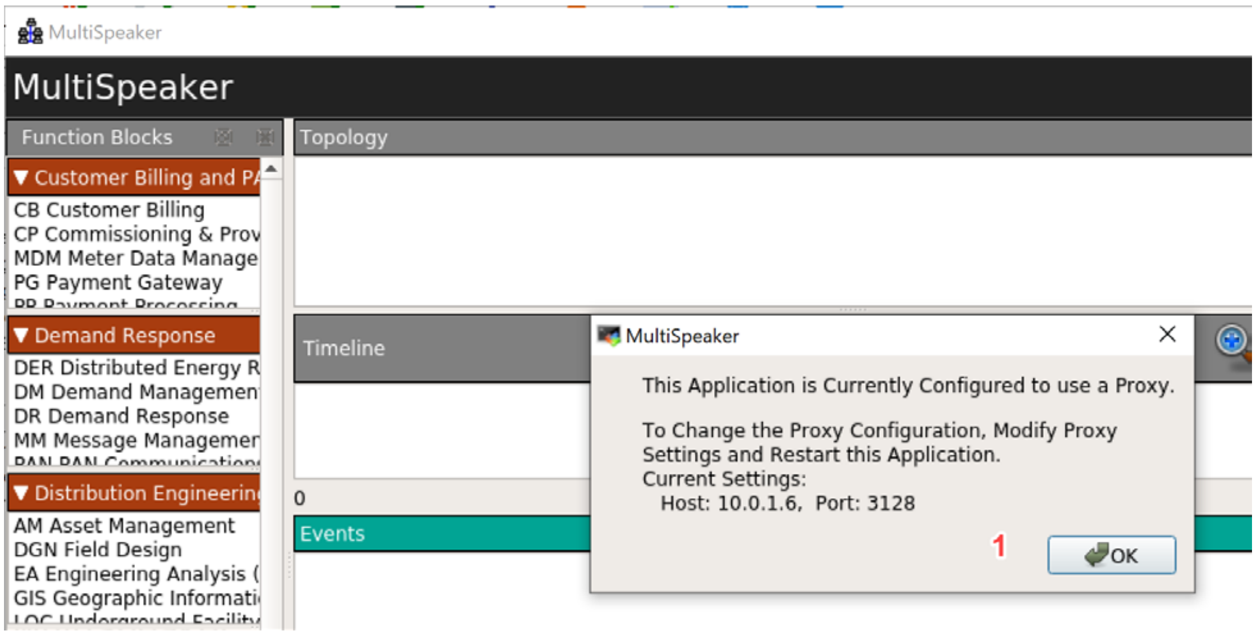
[1] https://github.com/pnnl/ms-speak/tree/Phase2/Multispeaker
[2] https://github.com/pnnl/ms-speak/blob/Phase2/Multispeaker/videos/MS-SPEAK-on-Azure.mp4
[3] https://mobaxterm.mobatek.net/

4. For listening on a specific port number.
5. Should be selected for enabling SSL encryption (currently not supported).
6. For clearing the logging data in the box below this button.
7. For closing the MultiSpeakerServer application.

A.2 MultiSpeaker Client

Once this application opens, you can find its network-of-computers icon in the Windows Taskbar, next to MobaXterm.

Figure 6 below shows the interface of MultiSpeaker with 'Function Blocks' in the left, 'Methods' on the right, and dedicated spaces in the middle for specifying 'Topology', 'Timeline' and 'Events'.



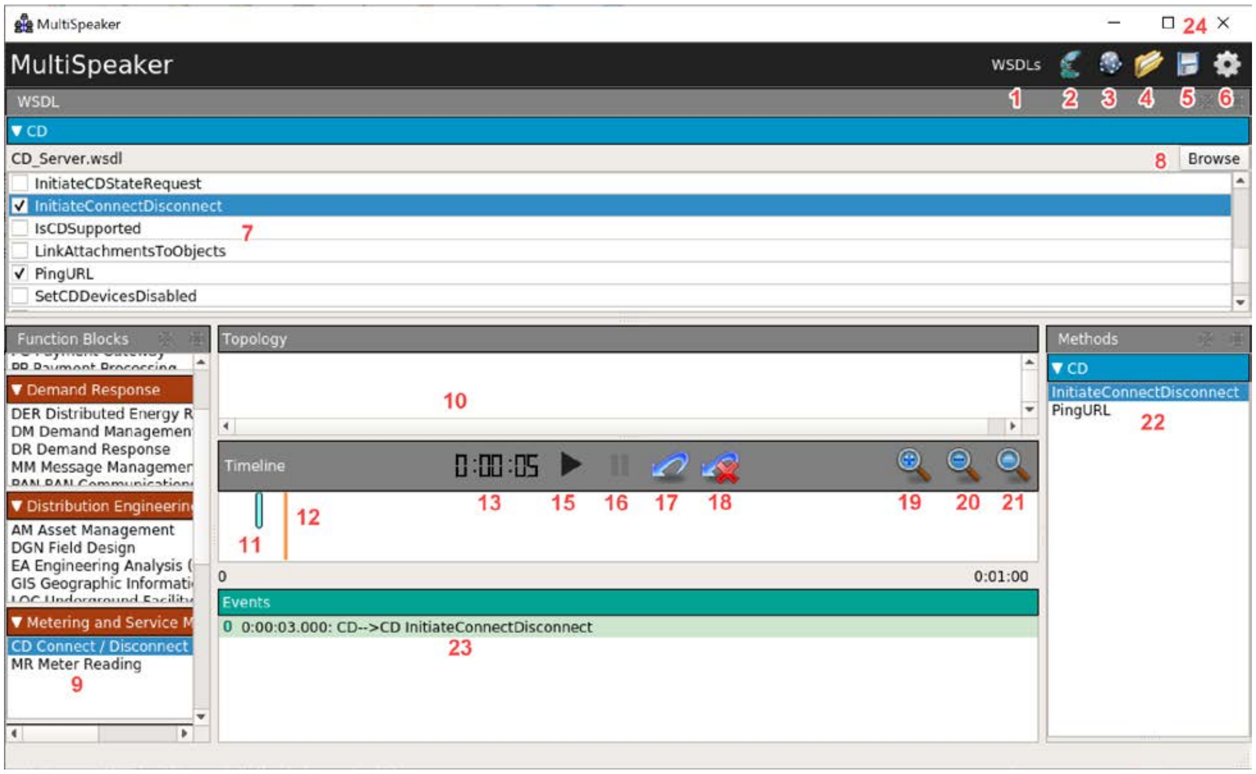
This Application is Currently Configured to use a Proxy.

To Change the Proxy Configuration, Modify Proxy Settings and Restart this Application.
Current Settings:
  Host: 10.0.1.6, Port: 3128

Figure 6. MultiSpeaker Client Interface Requires a Proxy Configuration

1. On starting the MultiSpeaker Client application, you will receive a prompt about configuring the application to use a proxy. Check the host address and port number indicated in the prompt and follow instructions for changing the proxy configuration, if the current settings need updating. Click “OK” on the prompt to proceed.

After dismissing this prompt, the full MultiSpeaker Client interface appears as in Figure 7.



```
0  0:00:03.000: CD-->CD InitiateConnectDisconnect
```

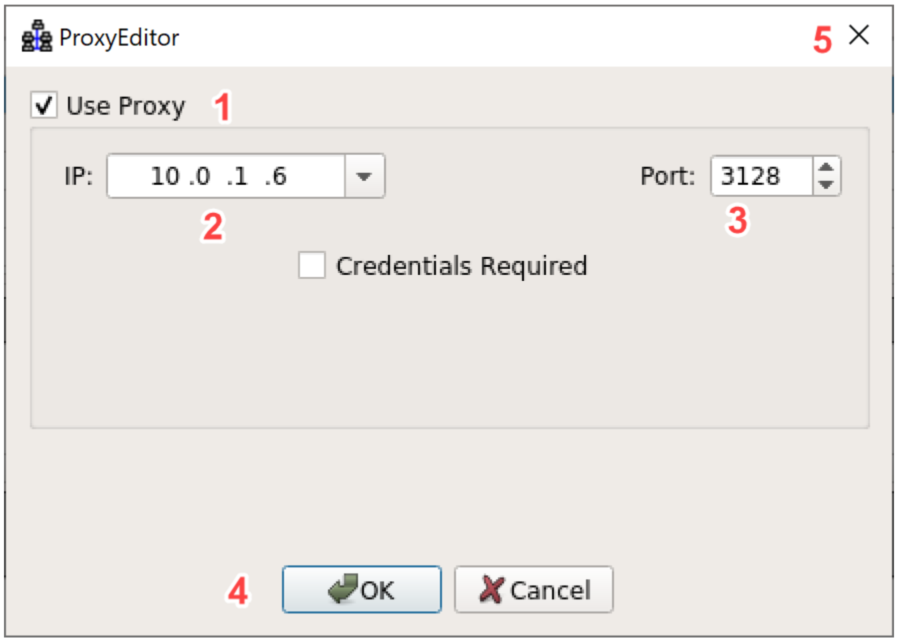
Figure 7. MultiSpeaker Client Interface for Developing Test Scenarios

1. For accessing/selecting the WSDL files. If the methods don't show up in the 'Methods' column in the right, then click on 'WSDLs' to select the appropriate WSDL files from 'CD_Server.wsdl' (see Figure 10).
2. ProxyEditor (see Figure 8). Click on the proxy editor (icon present above the 'Methods' column) and set up the proxy address.
3. HttpOutEditor (see Figure 9). Set up the HTTP IP and port address.
4. Select File option, for specifying previously saved scenario settings.
5. Save File option, for saving current scenario settings.
6. Version and build information about the MultiSpeaker client.
7. Check boxes to select the specific endpoint methods for the current WSDL.
8. For browsing to WSDL files for use.
9. Function Blocks tab lists the different endpoints by functional use. Select the function from the left 'Function Blocks' column. You can drag it to the 'Topology' space.
10. Topology space to create the topology by dragging the endpoints here from the Function Blocks.
11. After double-clicking the appropriate method from the 'Methods' column in the right (e.g., here we have selected on 'InitateConnectDisconnect'), an option will be provided to edit the request and response message related details (see Figure 11), which will lead to

creation of the timeline of the events. Each event is indicated in the timeline by a **cyan colored** vertical bar.

12. When the timeline is played, the orange vertical line moves as time progresses.
13. Current time in the timeline.
14. (not used).
15. "Play" button for starting or resuming the timeline of events. When the orange line (see 12.) crosses a cyan event mark (see 11.) the request associated with that event will be sent to the MultiSpeakerServer.
16. "Pause" button – temporarily stops the timeline from proceeding.
17. "Rewind" button to the timeline's beginning.
18. "Reset" button that clears the timeline.
19. For zooming in into the timeline.
20. For zooming out in the timeline.
21. Zooms back to the original timeline view.
22. The Methods tab on the right lists the various MultiSpeak methods. The appropriate method associated with the selected function needs to be double clicked for starting/creating a timeline.
23. List of the events in the timeline along with time, function and method details.
24. Closes the main window and application.

Upon clicking item 2 in Figure 7, the ProxyEditor appears as in Figure 8.



Use Proxy: ✓
IP: 10 .0 .1 .6
Port: 3128
Credentials Required: ☐

Figure 8. MultiSpeaker Client Proxy Editor

1. Enable the use of proxy address.
2. Input the appropriate proxy address here.
3. Input the appropriate port number here.
4. After updating the details click “OK” button to finish the ProxyEditor setup.

After clicking item 3 in Figure 7, the HttpOutEditor appears as in Figure 9.

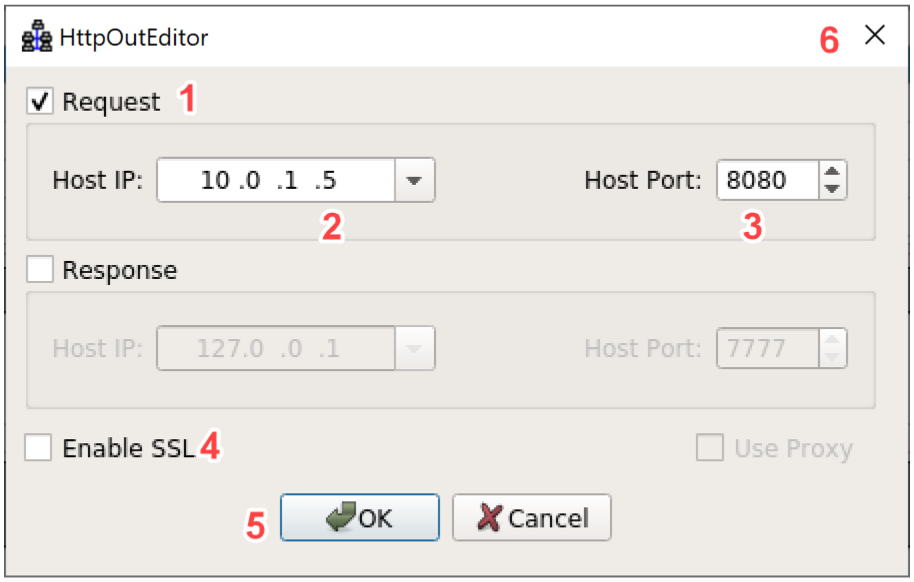
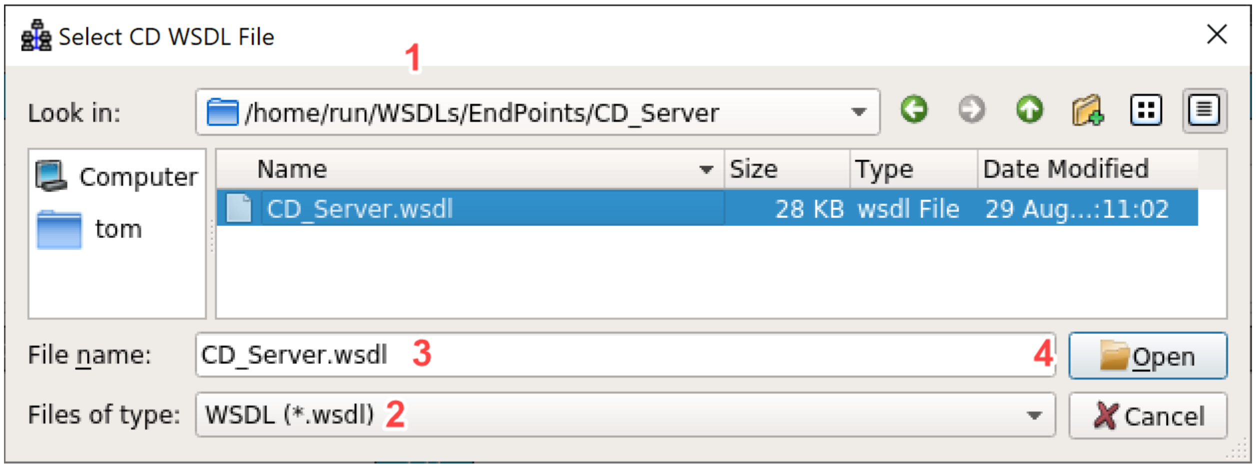


Figure 9. MultiSpeaker Client HTTP Out Editor

1. Enable the use of host IP address.
2. Input the appropriate host IP address here.
3. Input the appropriate host port number here.
4. Enable the use of SSL.
5. After updating the details click “OK” button to finish the HttpOutEditor setup.

After clicking item 1 in Figure 7, the WSDL File Open Dialog appears as in Figure 10.



/home/run/WSDLs/EndPoints/CD_Server

Name	Size	Type	Date Modified
CD_Server.wsdl	28 KB	wsdl File	29 Aug...:11:02

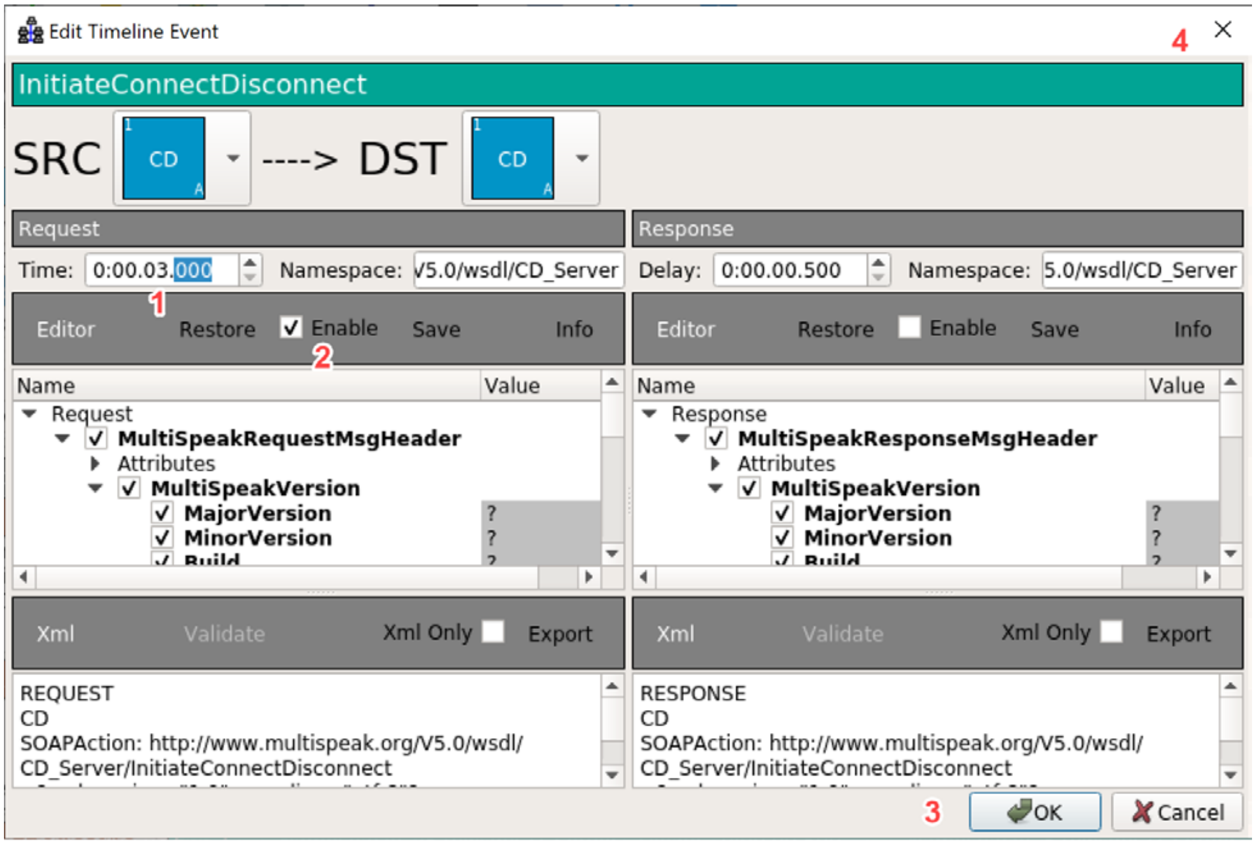
File name: CD_Server.wsdl

Files of type: WSDL (*.wsdl)

Figure 10. MultiSpeaker Client WSDL File Open Dialog

1. Navigate to the appropriate location for accessing the WSDL files.
2. Type of file should be set to .wsdl. It should come up by default.
3. Filename of the selected WSDL.
4. For displaying history of previously selected files.

After dragging a new event onto the timeline in Figure 7, the Edit Timeline Event dialog appears as in Figure 11. For MS-SPEAK, only a limited number of edits will be needed.



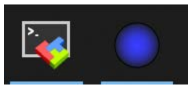
REQUEST
CD
SOAPAction: http://www.multispeak.org/V5.0/wsdl/
CD_Server/InitiateConnectDisconnect

RESPONSE
CD
SOAPAction: http://www.multispeak.org/V5.0/wsdl/
CD_Server/InitiateConnectDisconnect

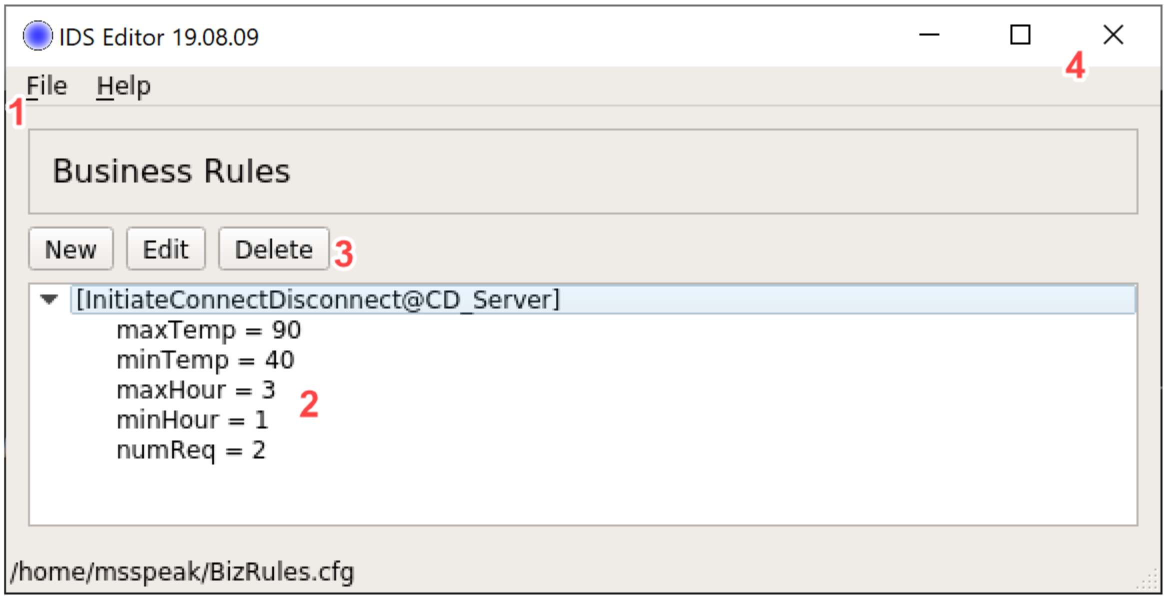
Figure 11. MultiSpeaker Client Interface to Edit Timeline Events

1. Set the time in the timeline when the request packet will be sent.
2. Always set this check box for MS-SPEAK. No other changes are needed.
3. Click OK to accept your edits.
4. Closes the dialog; clicking OK or Cancel is preferred.

A.3 IDS Business Rule Editor


Once this application opens, you can find its blue globe icon in the Windows Taskbar, next to MobaXterm.

This IdsEditor (see Figure 12) is used to configure business rules. It is possible to add, edit or delete business rules. The existing business rule editor serves as a simple example. The available source code can be customized to include more complex business rules.



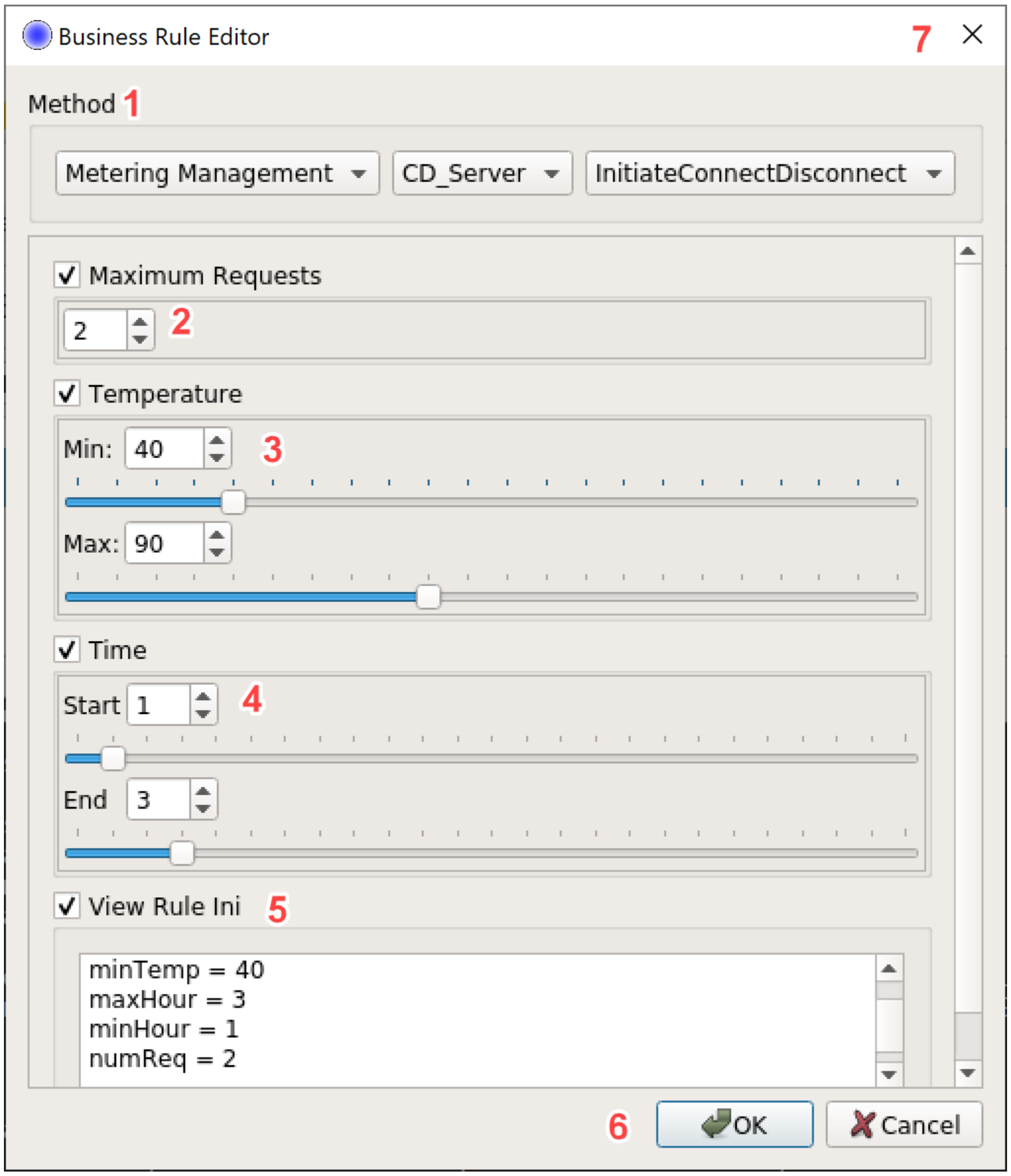
[InitiateConnectDisconnect@CD_Server]
maxTemp = 90
minTemp = 40
maxHour = 3
minHour = 1
numReq = 2

/home/msspeak/BizRules.cfg

Figure 12. IDS Editor Interface

1. File option for opening or saving.
2. List of the existing business rules. In the business rule editor, the following can be specified:
   a. minTemp: Minimum temperature allowed, above which remote disconnect can be initiated
   b. maxTemp: Maximum temperature allowed, below which remote disconnect can be initiated.
   c. minHour: Earliest hour in a day (formatted as just the rounded hour value of the hour in a 24-hour day) after which remote connect and disconnect can be initiated. Remote connects or disconnects cannot be initiated before this time on any day.
   d. maxHour: Latest hour in a day (formatted as just the rounded hour value of the hour in a 24-hour day) before which remote connect and disconnect can be initiated. Remote connects or disconnects cannot be initiated after this time on any day.
   e. numReq: Maximum number of requests that can be readily configured in a day.
3. Select whether you want to create new business rules, edit or delete the existing ones. On clicking “Edit” the Business Rule Editor window appears which has been showed below.

Upon clicking the Edit button, the dialog in Figure 13 will appear to edit these rules. (Note: they can also be edited outside the graphical user interface with a text editor.)



minTemp = 40
maxHour = 3
minHour = 1
numReq = 2

Figure 13. Business Rule Editing Dialog

1. Select the type of Method.
2. Enable by checking box, then select value of numReq.
3. Enable by checking box, then select values of minTemp and MaxTemp by dragging the seek bar or by typing the numbers in the appropriate fields.
4. Enable by checking box, then select values of minHour and MaxHour by dragging the seek bar or by typing the numbers in the appropriate fields.
5. Check this box to view the current business rule values in a list format below.
6. When editing is complete, select “OK” to finish.

PNNL-29393