



## SOAP: SCEPTRE ON A PLATTER

---

Sandia National Laboratories

SAND2020-84390

# User Guide

---

Prepared by:

Lisa Batsch-Smith (lbatsch@sandia.gov), Ryan Kao (rkao@sandia.gov)

---

SANDIA NATIONAL LABORATORIES

# SOAP: SCEPTRE On A Platter User Guide

---

© Sandia National Laboratories  
1515 Eubank Blvd SE  
Albuquerque, NM 87123



Exceptional service in the national interest

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

# Table of Contents

Welcome to SOAP .....	1
What is SCEPTRE and What Does It Do? .....	3
phēnix is Your Friend .....	4
What is included in SOAP? .....	7
IEEE 300 Bus System (Provider) Layer .....	7
SCEPTRE Relay Layer .....	8
Ignition SCADA System Layer .....	8
Control Things IO Testing VM.....	9
How to Run SOAP .....	10
Tools That Have Been Included .....	17
WIRESHARK .....	18
GRASSMARLIN.....	19
Ready to Give SOAP a Try?.....	21
Now Try These:.....	23
Use Case Ideas for SCEPTRE.....	25
Appendix.....	26
Relevant Accounts and Passwords.....	26
SOAP Network Diagram .....	27
SOAP Data Flow Diagram .....	28

## Welcome to SOAP

*“Tell me, I forget. Show me, I remember. Involve me, I understand.” – Chinese Proverb*

In order to better acquaint users with the SCEPTRE Industrial Control System (ICS) simulation platform, Sandia National Laboratories (SNL) has developed the SCEPTRE-On-A-Platter (SOAP) experiment range. SOAP is provided on a bootable media device that includes a pre-configured, sample SCEPTRE environment. This allows users to easily explore what SCEPTRE has to offer, from an immersive, functioning ICS simulation perspective. It also provides insight into the overall SCEPTRE structure so that users can better envision how SCEPTRE could best be used to support their particular mission. Overall, SOAP allows the user to better understand both the usage of and functionality contained within the SCEPTRE simulation environment.

Booting from the SOAP media device launches an interactive micro-grid environment with which the user may interact in the role of a power system operator. The simulated ICS environment also illustrates the capability within SCEPTRE to view system changes, such as when an operator command is executed, propagated through the system and the new system state is reflected on the Operator Station human machine interface (HMI). Additionally, tools have been provided to allow the user to monitor the simulated data flow, view the simulated network structure and capture packet data for analysis, as could be done in a field-implemented ICS.

The SOAP environment has been intentionally configured in a non-secure state, although the capability exists for it to be augmented with security features. Within SOAP, there are no firewalls or additional cybersecurity features installed or enabled. As a result, it provides a training environment for those who wish to learn vulnerability assessment or the impacts of applying cybersecurity components to a “functioning” ICS system, without any risk to physical systems or hardware.

Chapters 2 & 3 of this guide introduce the structure of the SCEPTRE simulation platform and the SOAP environment in order to introduce the components and how they combine to create the ICS simulation environment. Chapters 4 & 5 explain how to

start and interact with the simulation environment and the additional tools that have been provided for the user within the structure. Chapter 6 provides some activities to try as a system operator to illustrate the user interface and hopefully lead to a better understanding of the standard micro-grid operation. And Chapter 7 discusses some use cases for the SCEPTRE application that the user may find thought-provoking.

Welcome to SOAP!

## What is SCEPTRE and What Does It Do?

The SCEPTRE platform is at the heart of the SOAP demo. SCEPTRE is a simulation structure, developed at SNL, that creates an operationally realistic control system environment. Multiple pieces work in unison to represent the different components in an industrial control system. In addition to the full simulation capability, SCEPTRE also allows for hardware-in-the-loop (HIL) components to be fully integrated with the simulation environments. This allows for testing and evaluation of the component and its configuration as it would be installed and running in the actual operational environment. For our example SOAP experiment, we have not included any HIL devices.

SCEPTRE provides the framework to configure the TOPOLOGY required to simulate the system of interest. Components may be configured with internal protection schemes, control and reporting capabilities. A virtual network is established between these components, which execute on virtual machines (VMs) that provides the ability to view the communication between various system components. Therefore, although the networking is an emulated, software-defined structure, the ability to view, test, analyze and evaluate the data flow through the system is fully replicated and observable.

The overall SCEPTRE stack is shown on Figure 2.1. A specific instantiation of a SCEPTRE simulation that is configured and executed is referred to as an EXPERIMENT. The experiment is orchestrated by the phēnix management tool and its underlying libraries. These can be interacted with using phēnix's web interface.

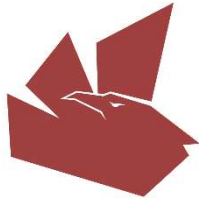


FIGURE 2.1 Components of the SCEPTRE stack.

Minimega is the tool that coordinates the VMs for each device into the working configuration. QEMU, in conjunction with KVM, is the hypervisor that is used within the structure to configure and launch the VM components. The VMs within the structure can take multiple forms. They can be general purpose hosts with configurations that have a full operating system and an industry standard control application (e.g., supervisory control and data acquisition (SCADA), distributed control system (DCS), etc.) or they can be a streamlined version that mimics the functionality of a protective relay or programmable logic controller (PLC).

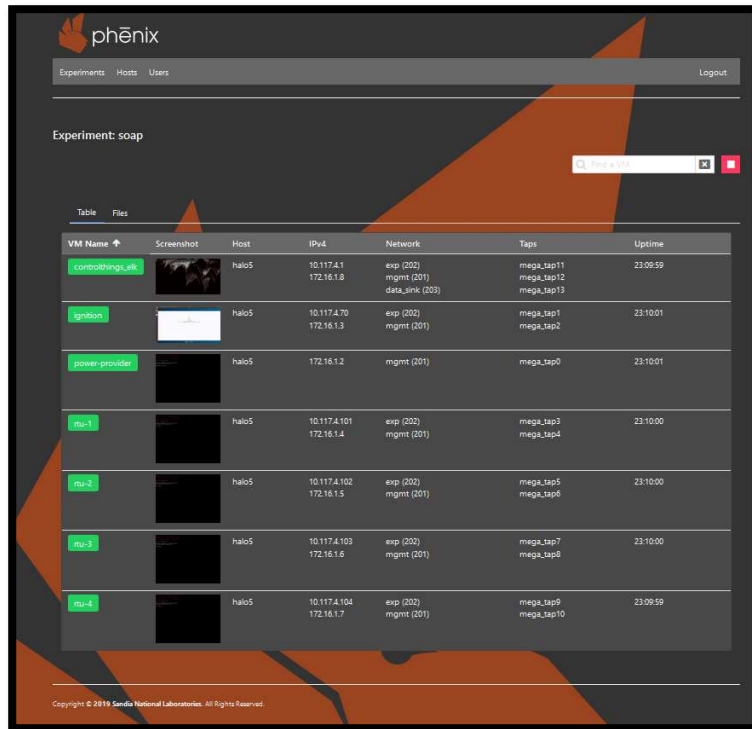
Documentation and more information on the various components of the SCEPTRE stack may be found here:

- minimega — <http://minimega.org/articles/>
- QEMU — <http://wiki.qemu.org/download/qemu-doc.html>
- KVM — <http://www.linux-kvm.org/page/Documents>



#### phēnix is Your Friend

When the user launches the SCEPTRE environment, interaction with the experiment and its elements happens within browser windows. The phēnix web interface provides the user with an overview of the virtual machines that are configured to function within the experiment, the virtual networks that the VMs are an element of, and the IP addresses for those connections. Figure 2.2 shows an example of viewing the SOAP experiment inside the phēnix web interface.



VM Name	Screenshot	Host	IPv4	Network	Taps	Uptime
controlthings_ek		halo5	10.117.4.1 172.16.1.8	exp (202) mgmt (201) data_sink (203)	mega_tap11 mega_tap12 mega_tap13	23:09:59
ignition		halo5	10.117.4.70 172.16.1.3	exp (202) mgmt (201)	mega_tap1 mega_tap2	23:10:01
power-provider		halo5	172.16.1.2	mgmt (201)	mega_tap0	23:10:01
mu-1		halo5	10.117.4.101 172.16.1.4	exp (202) mgmt (201)	mega_tap3 mega_tap4	23:10:00
mu-2		halo5	10.117.4.102 172.16.1.5	exp (202) mgmt (201)	mega_tap5 mega_tap6	23:10:00
mu-3		halo5	10.117.4.103 172.16.1.6	exp (202) mgmt (201)	mega_tap7 mega_tap8	23:10:00
mu-4		halo5	10.117.4.104 172.16.1.7	exp (202) mgmt (201)	mega_tap9 mega_tap10	23:09:59

FIGURE 2.2 SOAP experiment inside the phēnix web interface. Each of the elements in green are a separate VM and the IP addresses and networks associated with those VMs are listed in tabular format.

Each of the green boxes shows the name given to that VM during creation/configuration. The screenshot image is to further distinguish each VM to the user and allows direct interaction (i.e., console access) with a particular VM by clicking on the image. The host is the name of the physical machine on which each VM is running. The IP addresses are those assigned to each of the network interfaces defined for that particular VM. All devices will have communication on the management network, which is used for SCEPTRE coordination. The management network is distinct from the control system network (a.k.a., experiment network) and does not process ICS data transmission. Any devices that normally have communication on a physical ICS will have an IP address assigned to the experiment network. This is where the configured datalinks would communicate with the ICS protocols, such as Modbus or DNP3. Additional networks are created, as needed, depending on the configuration and usage of the experiment.

When the experiment is launched, all of the VMs will be booted and running as determined by the pre-configured starting state of the experiment. Phēnix gives the user an overview of the entire running system. This is different than viewing VMs in a hypervisor, where the machines are individually started and stopped. Clicking on the screenshot for an individual machine opens a new browser tab where the user can directly interact with the selected VM.

The minimega and KVM/QEMU functions are accessible for configuration purposes, but for the SOAP demonstration platform, these applications will not require any configuration work from the user.

The VMs that are configured to be ICS field devices – generally more streamlined devices than general purpose machines when installed in the field – are ran within SCEPTRE on VMs that mimic the dedicated functionality of a single device, such as a protective relay for an electrical system. These devices are configured from a generic Linux VM that runs the device simulation application called **SCEPTRE-ICS**. When a SCEPTRE-ICS device is added to the topology, it is configured to have communication with a simulation of the physical process (in this case, a power grid) and with the supervisory system such as SCADA or DCS. The communication with the physical process model occurs over the management network and is not data that would be observed through network monitoring on an actual ICS. This would be data collected by local metering and control commands that physically actuate system elements. The communication with the supervisory control system is configured to mimic the network traffic that would exist with the field device, such as Modbus, DNP3, BACnet, CANbus and other industry standard ICS protocols.

The VM that is running the model of the physical process is referred to as the PROVIDER. This is a stand-alone simulation application that receives system



commands from the SCEPTRE-ICS devices, calculates the resulting state and provides the current metered and monitored conditions (e.g., currents, voltages, breaker positions) as output values. The Provider is typically an application that is well-known for power-system stability solution, such as PowerWorld, Matlab, PyPower, etc. The fidelity level of the provider determines how closely it will simulate the results of the physical system. Therefore, the level of realism that is required by the user for the purposes of their specific experiment needs to be clearly defined at the initial SCEPTRE design stage, so that the appropriate provider, and time for development, are determined correctly.

## What is included in SOAP?

Figure 3.1 illustrates the one-line structure of the microgrid that is being modeled by the SOAP architecture in our specific experiment, if it was designed as a physical system. This structure is a section taken from the standard IEEE 300 bus architecture. This standard structure is available as a pre-built configuration case file that can be loaded into PyPower. The entire case file runs in PyPower, but the portion displayed on the operator control interface is a small portion of the experiment that we have given names and functionality to through the SCEPTRE configuration. The SCEPTRE virtual architecture within SOAP provides you with the ability to interact with four (4) buses (defined within the IEEE model as buses 242-245). These buses have generators, loads and branch (line) connections to the other buses. Additionally, two of the buses have transformers defined between them. All of these features are shown as independent items within SOAP.

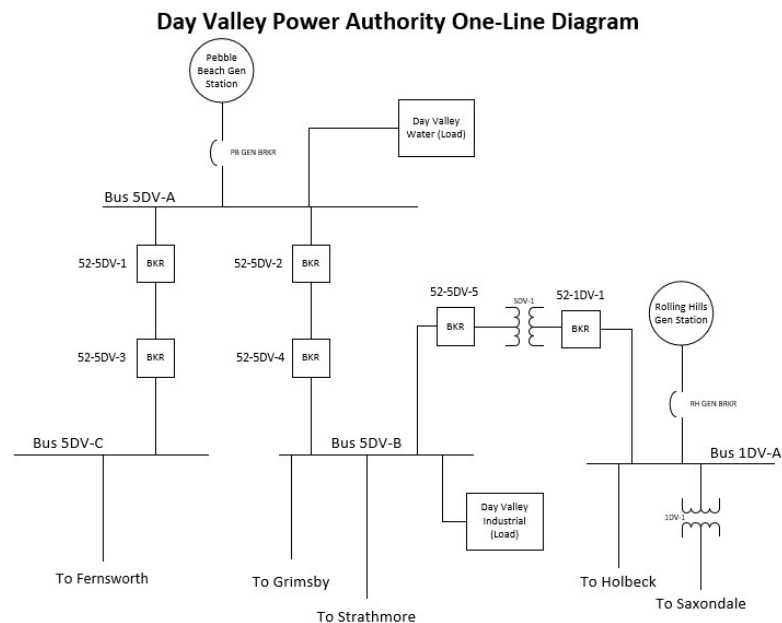


FIGURE 3.1 One-line diagram for the SOAP architecture.

### IEEE 300 Bus System (Provider) Layer

The standard IEEE 300 bus system is being utilized as the defined configuration for the provider. SOAP is utilizing PyPower, which is an open-source power flow solver.

Information on this software application may be found here:  
<https://github.com/rwl/PYPOWER>.

Within SOAP, the entire IEEE 300 bus system is being solved for a steady-state solution during each execution cycle using PyPower in the Provider VM. However, each field device relay, RTU, etc., is launched as a separate VM, referred to as a SCEPTRE-ICS device and consumes a portion of the system resources. Since SOAP has been designed to run on limited (not server-level) computing resources, the number of field devices that have been configured for operator interaction has been limited. For full-sized system simulations that are run on server, computing cluster or cloud-based installations, the number of devices that may be interacted with are not limited in this manner.

**In SOAP, all relays communicate to the Operator HMI via DNP3 protocol.**

#### **SCEPTRE Relay Layer**

There are individual, simulated relays (or remote terminal units, RTUs) that have been created for each of the controllable elements (i.e., breakers). Each one of these devices has its own IP address and communicates with the Operator HMI providing the status for different variables, such as voltage, power and breaker status. The PyPower simulation receives the commands from these devices in order to modify the state of the power system by changing the breaker position.

From the list provided in the phoenix web interface, you can see each of these listed as rtu1-rtu4. Although there are more breakers than this in the SOAP power system, the breakers that are at the beginning and end of each transmission line have been paired to isolate or re-energize that transmission line by clicking either one of the paired relays (such as 52-5DV-2 and 52-5DV-4). This was done to both conserve resources within the SOAP architecture and to work with the constraints of the IEEE 300-bus case file that was pre-developed for PyPower. When considering SCEPTRE applications, keep in mind that each field device controller (relay, RTU, etc.) will require an additional SCEPTRE-ICS VM for that application and the additional resources required for that VM.

#### **Ignition SCADA System Layer**

Ignition SCADA<sup>1</sup> is used as the user interface for SOAP. This is a commercially available product that aggregates the data from the virtual relays and provides both a means for the operator to view the local system state and to request state changes to the loads, generation and transmission line breakers.

The sample HMI requires no additional configuration for the purposes of the SOAP demo. However, if more information is desired about the Ignition SCADA suite, you can find more information here:

<sup>1</sup><https://inductiveautomation.com/>.

### Red-Team Note:

Inductive Automation has provided a patch to their system that was discussed in ICS-CERT ICS Advisory (ICSA-20-147-01). This patch is not applied in SOAP. Should a user wish to evaluate potential impacts of an unpatched, known vulnerability in a simulated ICS environment, more information on this may be found here:

<https://www.us-cert.gov/ics/advisories/icsa-20-147-01>



### Control Things IO Testing VM

A VM with the ControlThings.io<sup>2</sup> environment has been added to the experiment to provide an integrated testing suite so that you may analyze the SOAP environment in the same way that you would analyze a physical ICS environment. More information on the tools and features available within the ControlThings.io platform can be found here:

<sup>2</sup><https://www.controlthings.io/>

This VM has been added to the SOAP environment with the ‘\_elk’ extension added to the VM name, indicating to SCEPTRE that a SPAN port for each of the virtual switches should be added so all data on the network may be observed by this device. This is important for running the tools that collect and analyze all ICS network traffic. SPAN ports should be configured manually on hardware switches if an external test machine is incorporated into a SCEPTRE experiment.

Once the VM has booted, you will be automatically logged in as user "**control**". Direct root login has been disabled., but you can login as root by opening a terminal and running "**sudo -s**" with the password "**things**".

## How to Run SOAP

SOAP has been designed to run on a standard business workstation. A computer with a multi-core processor, integrated graphics card and a minimum of 16GB of RAM is recommended. The system also requires using UEFI firmware mode instead of BIOS mode for booting purposes.

The bootable media provided has been encrypted. When inserted into the host machine, the protective password will need to be entered for the encryption protection that has been installed on the device. This password is:

Password: 1qaz@WSX3edc\$RFV

After entering the password, SOAP is automatically launched, and the user's computer will boot into an Ubuntu Linux environment. This is the head node for the overall experiment management. This will require a login for the phēnix web interface as shown in Figure 4.1.

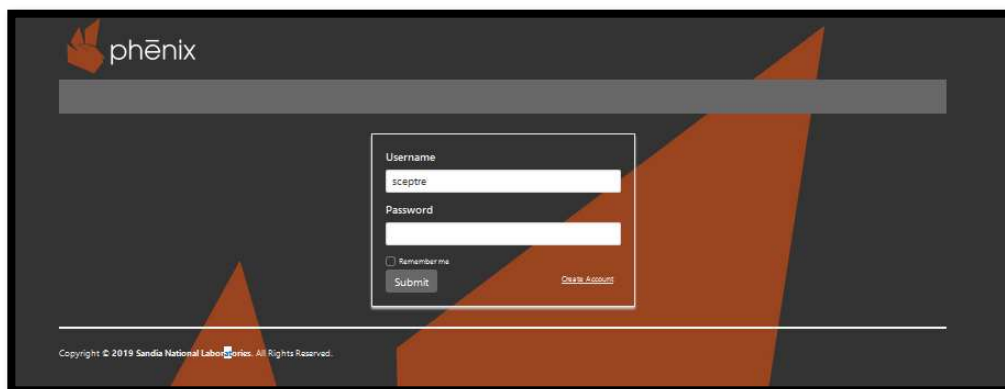


FIGURE 4.1: The phēnix web interface login screen on the head node.

The login credentials for SOAP are:

User: sceptre  
Password: soap

Note: These are case-sensitive and should all be lower-case.

Upon login you will be directed to the list of available experiments, as shown in Figure 4.2.

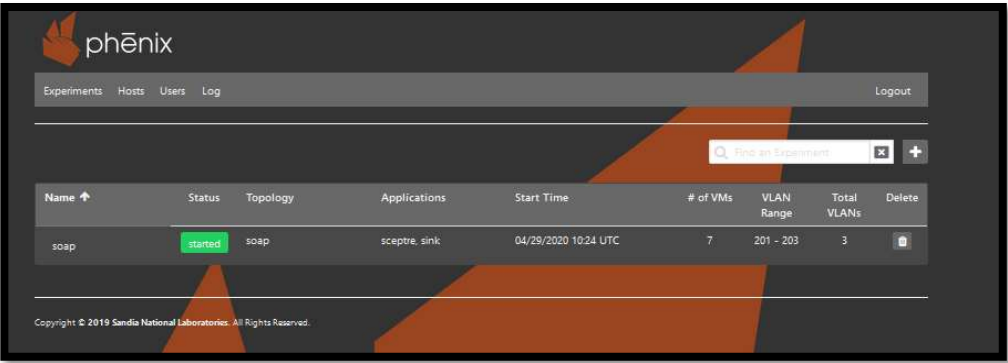


FIGURE 4.2 Experiment list for SOAP.

There is only a single experiment available within SOAP, named soap. The status shown in Figure 4.2 indicates that the experiment has started and is ready. To stop the experiment, click on that green “started” box. This will launch a conversation window that will allow you to stop the experiment. Selecting the red “stop” button, as shown in Figure 4.3 will stop the soap experiment.



FIGURE 4.3 Conversation window for stopping the SOAP experiment.

When the experiment list returns, you will notice that the status box is now red and reads “stopped”, as shown in Figure 4.4.



FIGURE 4.4 SOAP experiment showing STOPPED status.

To start the SOAP experiment once again, click on the “stopped status button, and this will launch the conversation window that asks if you wish to start the SOAP experiment, as shown in Figure 4.5.

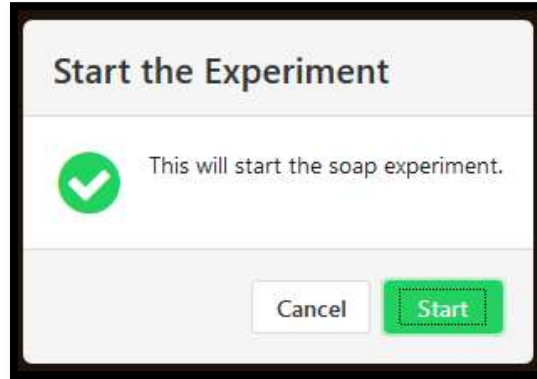


FIGURE 4.5 Conversation window for starting the SOAP experiment.

The starting process requires more time to accomplish than the stopping process. As a result, for a brief moment, the status button within the experiment list window will change to an amber colored status bar that will provide information on the starting process. Once the starting process is 100% complete, the status button will again change to a green “started” button, as shown in Figure 4.6.

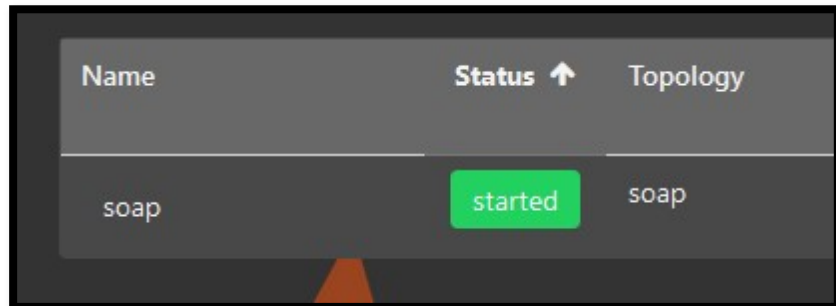


FIGURE 4.6 SOAP experiment showing STARTED status.

This process is discussed here in detail due to the fact that this procedure of restarting the experiment will be required when running operations, if a change is made that result in blacking out the system, as discussed in Chapter 6.

Once the status is confirmed to be “started”, the experiment is then formally “launched” for user interaction by clicking on the name of the experiment, which in our case is “soap”. This area, when highlighted, will change to a deeper gray, indicating it is selected. Clicking on the darker gray area will switch to the phēnix experiment VM info tab that will enumerate all of the VMs associated with SOAP, as shown in Figure 4.7.

The complete topology consists of the seven (7) VMs listed there:

- controlthings\_elk: Toolkit machine
- ignition: Data aggregation and operator HMI interface for monitoring and control
- power-provider: The PyPower simulation VM
- rtu-1 through rtu-4: Relay devices for control and monitoring of “field devices”

VM Name	Screenshot	Host	IPv4	Network	Taps	Uptime
controlthings_elk		halo5	10.117.4.1 172.16.1.8	exp (202) mgmt (201) data_sink (203)	mega_tap11 mega_tap12 mega_tap13	23:09:59
ignition		halo5	10.117.4.70 172.16.1.3	exp (202) mgmt (201)	mega_tap1 mega_tap2	23:10:01
power-provider		halo5	172.16.1.2	mgmt (201)	mega_tap0	23:10:01
rtu-1		halo5	10.117.4.101 172.16.1.4	exp (202) mgmt (201)	mega_tap3 mega_tap4	23:10:00
rtu-2		halo5	10.117.4.102 172.16.1.5	exp (202) mgmt (201)	mega_tap5 mega_tap6	23:10:00
rtu-3		halo5	10.117.4.103 172.16.1.6	exp (202) mgmt (201)	mega_tap7 mega_tap8	23:10:00
rtu-4		halo5	10.117.4.104 172.16.1.7	exp (202) mgmt (201)	mega_tap9 mega_tap10	23:09:59

FIGURE .4.7: SOAP VM info tab in phēnix.

Halo5, as the host, is the name of the machine on which the experiment VMs are running. The IP addresses are listed for each of the networks. Exp (202) is the ICS communication network. The mgmt (201) network is for managing the SCEPTRE environment. The data\_sink (203) is the created SPAN port taps for data collection and analysis by the controlthings\_elk toolkit VM.



Figure 4.8 shows the screenshot for the individual field-device VMs. Although there is a login prompt, it is not necessary to interact with the VM to run the experiment. The communication with the relays will already be established with the initial experiment start.

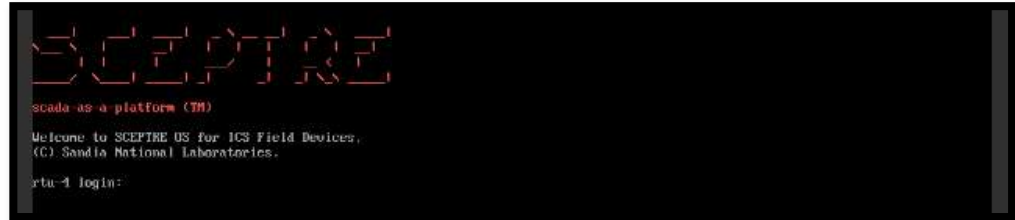


FIGURE 4.8: VM for an individual SOAP Field Device relay.

Clicking on the screenshot of the ignition VM will launch a new tab showing the desktop as shown in Figure 4.9. The Ignition software is started using the Ignition Vision Client Launcher. SOAP will be the only application listed for use.

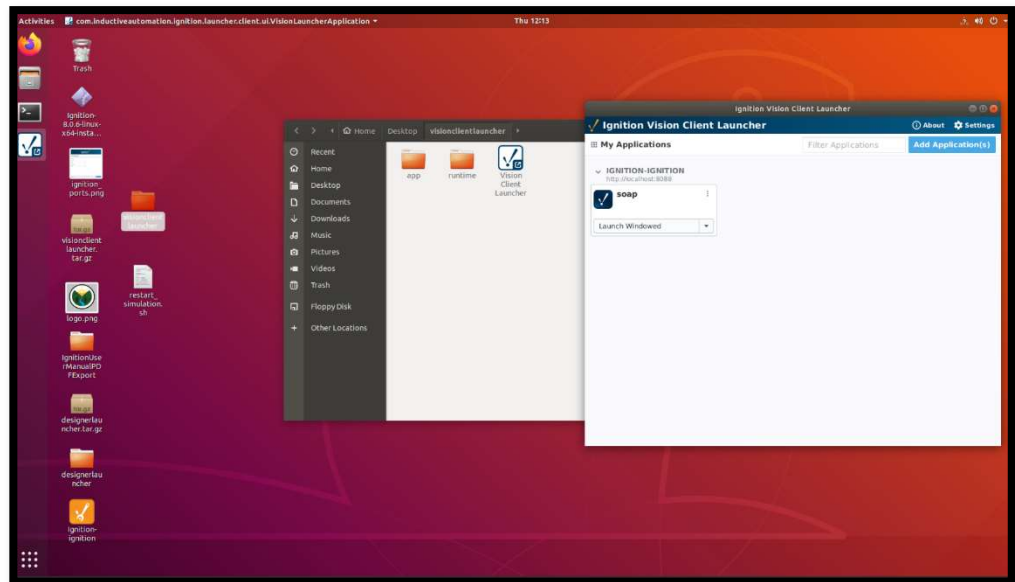


FIGURE 4.9 Ignition VM Desktop image

Once the Ignition software is launched, you will be presented with the login screen for the SOAP project as shown in Figure 4.10. The login credentials here are:

Username: admin  
Password: admin

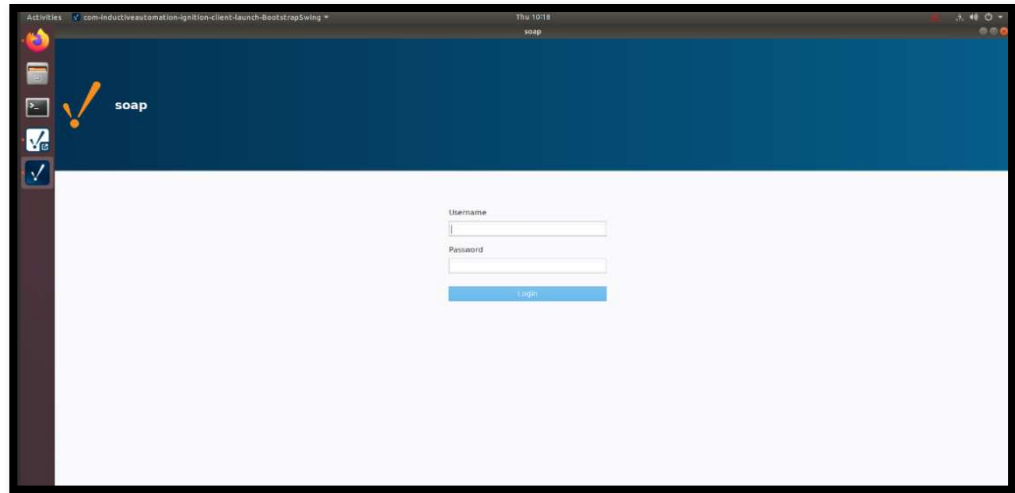


FIGURE 4.10 Login screen for the Ignition SCADA software

For the last step in launching the Ignition SOAP project for use is confirming the status of the Ignition license. The trial Ignition license included with SOAP is valid for 2 hours (unlimited restarts). If the trial license is expired, it will appear as an orange notification banner at the top of the webpage (<http://localhost:8088/>), as shown in Figure 4.11. Signing in with the username and password will allow the user to activate the temporary license via a selection button on this banner.

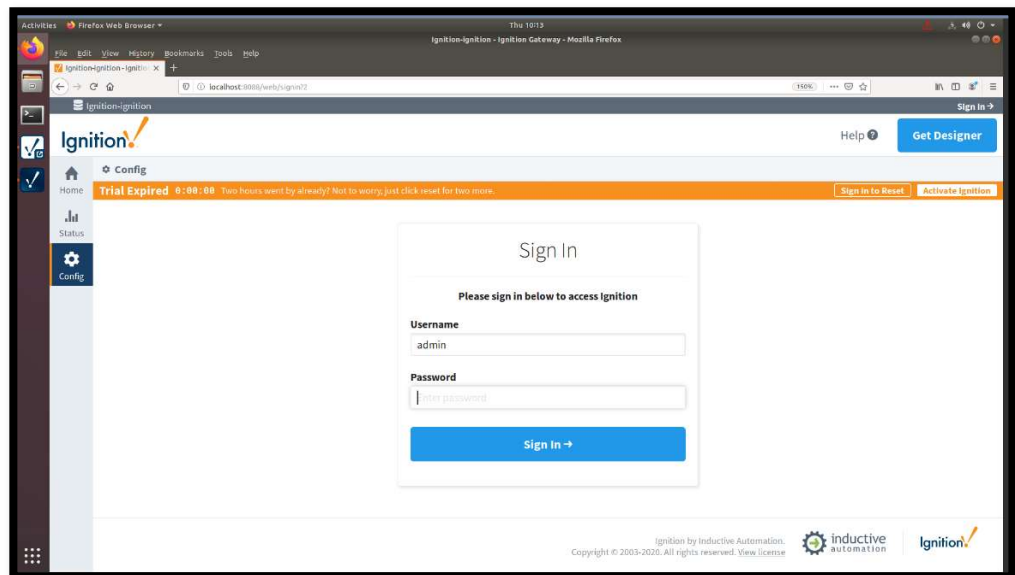


FIGURE 4.11 Login screen showing trial license expiration in orange

Once the trial license is active, the banner will turn green and have a timer counting down with how much time is remaining, as shown in Figure 4.12.



FIGURE 4.12 Ignition software with active trial license.

The operator interface can now be launched and you can use the SOAP environment as a power system operator.

In order to have a view into the structure and communication that is being simulated within SOAP, a VM with the ControlThings.io platform has been installed. The developers describe the ControlThings IO platform as “A Linux distribution to aid in the cyber security assessment and penetration testing of Industrial Control Systems (ICS), including SCADA, DCS, IoT, IIoT systems, field devices and field buses.” Below is a discussion of two of the tools that will be exceptionally useful for understanding the SOAP structure: Wireshark and GrassMarlin. If you would like to learn more about this toolset, information can be found [here](#):

Clicking on the “controlthings\_elk” VM screenshot in phēnix will open this VM in a new browser tab with what appears to be a blank desktop. Clicking on the “Activities” area in the upper left-hand corner will show the toolbar. From the toolbar, select the “Show Applications” button in the bottom left (nine dots icon) which will show all of the applications that are available in the ControlThings application. This listing is shown in Figure 5.1.



17

## WIRESHARK

Wireshark is a network packet analyzer tool that is nearly universal in any network professional's toolkit. This tool is used to verify communication between network nodes, determine the type of communication occurring and decode and read messages. Per the developers, the uses of Wireshark include:

- Network administrators use it to troubleshoot network problems
- Network security engineers use it to examine security problems
- QA engineers use it to verify network applications
- Developers use it to debug protocol implementations
- People use it to learn network protocol internals

All of these functions may be utilized within the SOAP environment. More information on using Wireshark may be found here:

[https://www.Wireshark.org/docs/wsug\\_html\\_chunked/](https://www.Wireshark.org/docs/wsug_html_chunked/)

To get started, network communication can be captured and stored in .pcap file format. To view network communication and store it as a .pcap file in SOAP, Wireshark has to be monitoring the correct network. Remember that there are multiple software defined networks within SOAP and the network(s) that each device is connected to is shown in that device's information block in phēnix. When Wireshark is launched, all of the available networks are shown in the configuration screen. To view the ICS data within SOAP, we must configure the Wireshark application to observe the communication on the "enpls3" interface. This interface listens to the aggregate of all experiment traffic in the environment. This selection is highlighted in Figure 5.2.

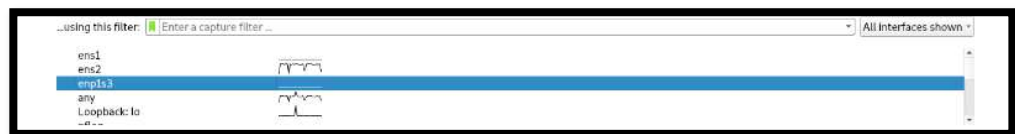


FIGURE 5.2 Selecting the "enpls3" interface to observe ICS communication within Wireshark.

Once the correct options are selected, you can start the Wireshark data capture by selecting the "Start" icon (🔵) and the main window will begin to populate with the observed communication. This data collection will appear as a series of rows that specify the sending IP address, receiving IP address, protocol used for the message, and other details as shown in Figure 5.3.

After collecting data for a period of time, stop the data collection using the "Stop" icon (🔴) and then save the collected data by selecting the "Save As" option from the "File" menu. Save the capture with any name you like, but remember it, because we will be using that data capture with the next tool, GrassMarlin.

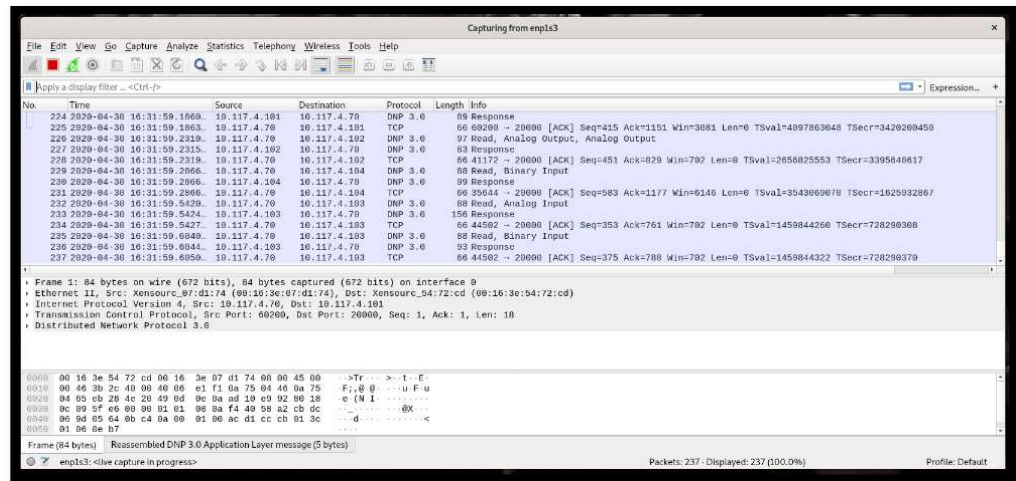


FIGURE 5.3 Sample Wireshark display showing the SOAP ICS Communication.

Note the DNP3 protocol listed as the communication between the relays and the Ignition SCADA server. This is as expected, since all SOAP relays were configured to only use the DNP3 protocol.

## GRASSMARLIN

This tool was developed to assist in providing situational awareness for an ICS. The full capabilities of this tool are discussed in the GrassMarlin User Guide, which may be downloaded here:

<https://github.com/nsacyber/GRASSMARLIN/blob/master/GRASSMARLIN%20User%20Guide.pdf>

The feature within GrassMarlin we discuss here is the ability to provide the user with a visualization of the network, based on the communication observed on the network. We can do this in an “offline” manner with the .pcap file that we saved using Wireshark. Open GrassMarlin and use the “Add Files” button to import a stored .pcap file, as shown in Figure 5.4.

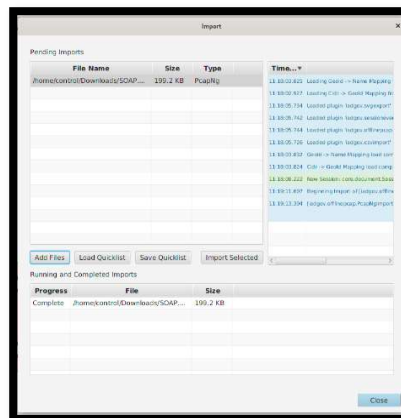


FIGURE 5.2 Interface for GrassMarlin Import window.

When this file is analyzed, a graphical representation of the devices located on the SOAP network is built, based on the observed communication between the devices. An example of this network graph is shown in Figure 5.4.

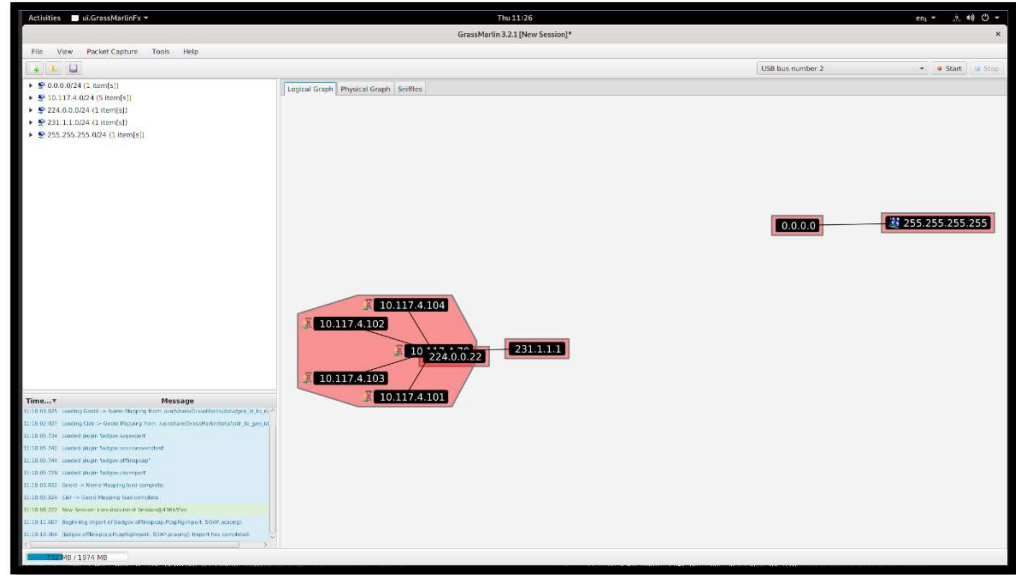


FIGURE 5.3 Network topology constructed from communication analysis by GrassMarlin for the SOAP architecture.

This graph provides insight into the devices present on the network and the communication pathways. As can be seen from the list above 10.117.4.102 (rtu-2) is communicating with 10.117.4.70 (ignition). Rtu-3 (10.117.4.103) is also communicating with ignition (10.117.4.70). This makes sense that the field device relays are updating the SCADA server with current system information. Note that 10.117.4.102 (rtu-2) is not communicating with 10.117.4.103 (rtu-3) even though they are within the same network boundary. Communication between relays is not normal communication within a SCADA system and is accurately represented in the SOAP communication structure.

## Ready to Give SOAP a Try?

In order to truly understand both the operation and the potential uses for SCEPTRE, SOAP provides hands-on experience. Within SOAP, the user can take on the roles of:

- a power system operator
- network communications analyst
- cybersecurity professional

all within the training environment.

In order to make the most of the sample simulation presented by SOAP, it is important to learn the role of the power system operator, within the limitations of SOAP. The small portion of the IEEE 300 Bus system that has been implemented within SOAP is represented within the Ignition software as a small grid that allows for interaction.



**Day Valley Power  
Authority**

Welcome to your new role as the power system operator for Day Valley Power Authority. In this role, you have the ability to open and close transmission line breakers and request changes in generation and loads. Clicking on the ignition VM screenshot in phoenix will open the Ignition SCADA VM in a new browser tab. The Day Valley Power Authority Operator HMI will be displayed as shown in Figure 6.1.

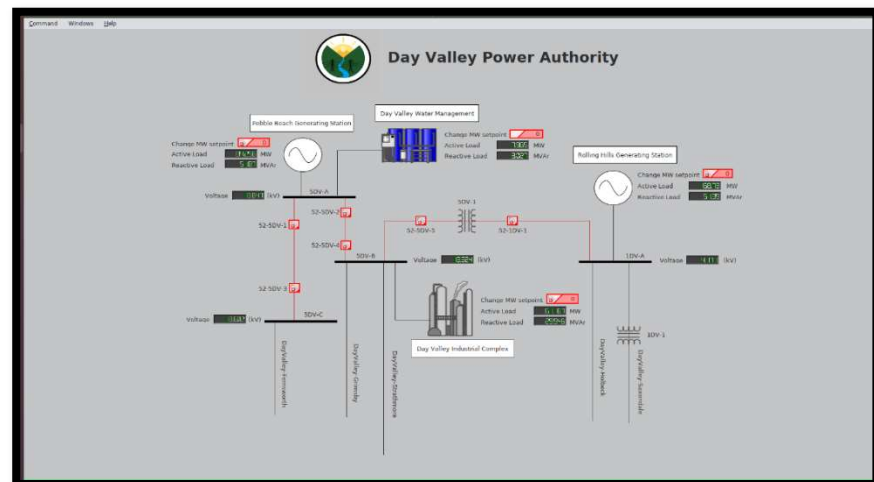


FIGURE 6.1: The Day Valley Power Authority Operator HMI Display



The values that are displayed are updated as the data is received from the virtual field devices. The elements that appear **RED** are the elements that may be interacted with by the system operator.

The PyPower provider will continue to calculate real-time stability analysis of the conditions within the power system. Some of the loads and generators have restrictions placed on the values that you may enter in order to keep the system stable. However, it is still possible to put the system into conditions that will result in the system becoming unstable and fail to solve. If actions taken by the operator place the system in this condition, the system will blackout, as shown in Figure 6.2. The simulation will need to be restarted.

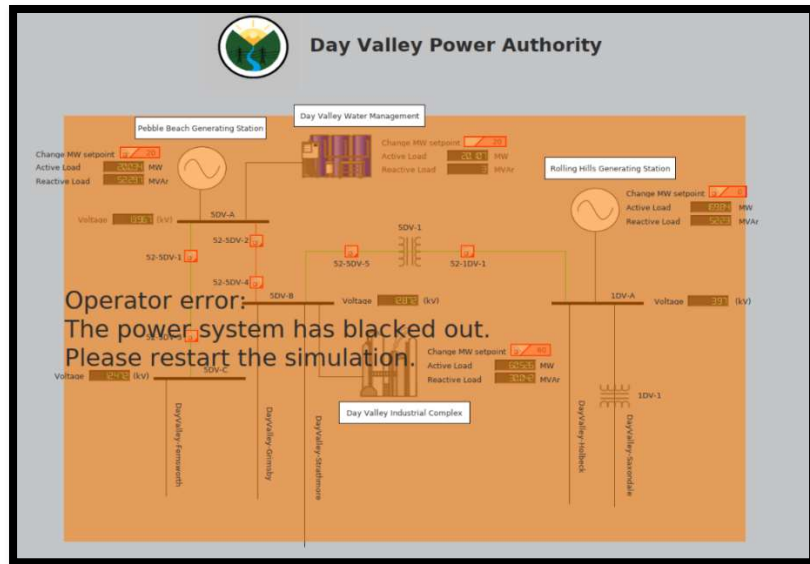


FIGURE 6.2 One-line diagram for the SOAP architecture.

```

ubuntu@ignition: ~/Desktop
File Edit View Search Terminal Help
ubuntu@ignition:~/Desktop$ ls
designerlauncher IgnitionUserManualPDFExport
designerlauncher.tar.gz logo.png
Ignition-8.0.6-linux-x64-installer.run restart_simulation.sh
Ignition-ignition.desktop visionclientlauncher
ignition_ports.png visionclientlauncher.tar.gz
ubuntu@ignition:~/Desktop$ cat restart_simulation.sh
#!/bin/bash
echo password is 5iaSd3te
ssh root@172.16.1.2 'pybennu-power-solver restart -c /etc/sceptre/config.ini -e
power-solver -d'
ubuntu@ignition:~/Desktop$

```

FIGURE 6.3 Simulator restart script on ignition VM

The user can restart the power simulation by running the bash script located on the desktop of the ignition VM. The script performs a command over ssh on the provider VM and requires the user to provide the root password, “**SiaSd3te**”. Once the system is restarted, operational actions in the HMI will be available again.

**Now Try These:**

To help familiarize you with operations within the Day Valley Power Authority system, below are a few activities to observe the changes in the system that can occur by operator action:

Activity 1: An increase in power is needed at the Industrial Complex. They have requested that you change load limit at the complex to 100MW. The plants at the complex will immediately load the grid with additional machinery.

Prior to changing the load limit, observe the output of the two generation stations. Once the load is increased, observe how much the output of the generators changed. Where is the power being supplied from?

Activity 2: The line between 52-5DV-2 and 52-5DV-4 needs to be taken down for repair work. Open these breakers to allow the repair work to proceed.

Does the reactive load at Pebble Beach Generating Station change? Why would this happen/not happen? What other effects can you notice within the system without this support line present?

Activity 3: 5DV-1 requires maintenance. The power needs to be isolated from this transformer by opening the surrounding breakers. They are joined, so if you open one, the other one opens. After the transformer is isolated, lower the load request to Rolling Hills generating station to 150MW. The Industrial Complex has requested you raise the load limit to 75MW.

After these changes, observe what has happened to the voltage on bus 5DV-1. What has caused this change? Would it be wise to put the transformer back in service under these conditions?

Activity 4: Fernsworth calls to request the connecting line be opened to assist with re-stabilization of their local grid. Open 52-5DV-1 and 52-5DV-3 to accomplish this.

What is the difference now in the voltages between Bus 5DV-A and 5DV-C.? Upon receiving confirmation that the stability situation was corrected, reclose the breakers. What is the difference in the bus voltages now? Why would this occur?

After learning how to change the system and what to expect, now you can utilize the network tools, discussed in Chapter 5, to monitor the network and the communication that requests these actions to take place.

## Use Case Ideas for SCEPTRE

The intention of the SOAP demonstration experiment is not only to familiarize a user with the structure and operation of the SCEPTRE platform, but to also spark ideas for how this powerful simulation platform could be used for your particular applications. With a better understanding of the SCEPTRE platform, it is easy to see how it could be used for the following use-cases:



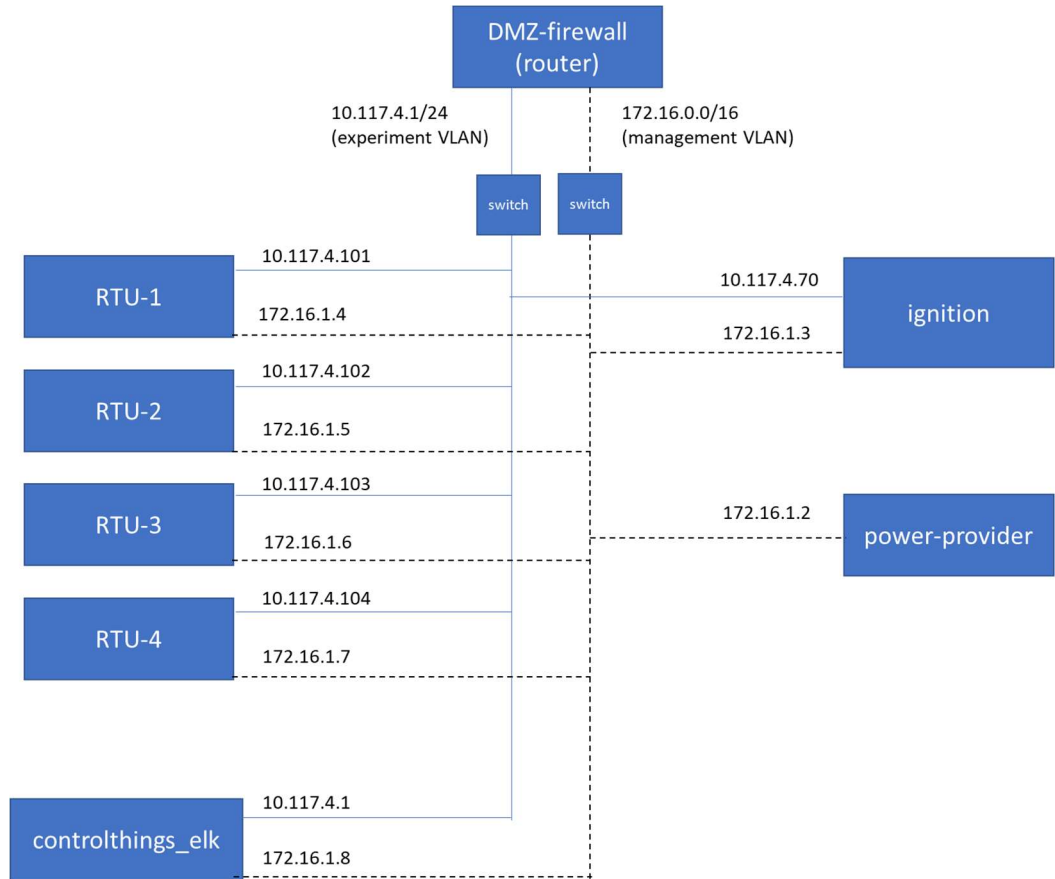
- 1.** Develop a model of a power system that would allow users to simulate possible user-errors in order to evaluate the resilience of the system, simulate possible results of these errors and design configuration improvements for safer operation.
- 2.** Design a power system that will allow for better training on system troubleshooting tools (i.e., Wireshark, etc.) for a deeper understanding of the way the tools may be used and the possible unintended impacts those tools could have on the overall system in a safe environment.
- 3.** Develop a model of a system and apply cyber-security features for testing their efficacy prior to system installation.
- 4.** Apply patches to devices within the simulation structure before applying them to the physical systems to ensure there are no unforeseen operational impacts.
- 5.** Replicate an existing power system and train individuals on pen-testing techniques without risk to hardware.

...and many more. We look forward to hearing how you envision the SCEPTRE environment will support your missions!

## Relevant Accounts and Passwords

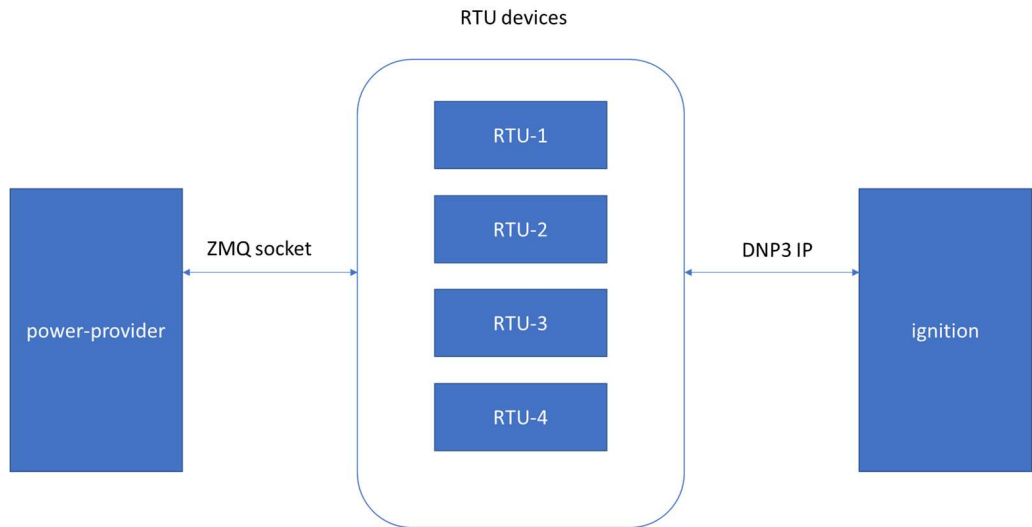
PURPOSE	USERNAME	PASSWORD
Unlocking SOAP drive upon boot	N/A	1qaz@WSX3edc\$RFV
rtu-1 through rtu-4 login	root	SiaSd3te
phēnix web page login	sceptre	soap
controlthings_elk VM	control	things
ignition Vision Client and control panel webpage ( <a href="http://localhost:8088">http://localhost:8088</a> )	admin	admin

## SOAP Network Diagram



- experiment VLAN
  - Solid lines represent connections of hosts to the experiment VLAN
  - Over this VLAN, ignition communicates to the RTUs over DNP3 IP protocol to query data points
- management VLAN
  - Black, dotted lines represent connections to the management VLAN
  - power-provider communicates to RTUs via ZMQ sockets, providing current state of the 300 bus power system

## SOAP Data Flow Diagram



- power-provider runs the 300 bus system simulation and constantly publishes the system state to RTUs
- RTUs use this state information to update DNP3 data points they are configured for
- ignition reads/writes to data points of the RTUs via DNP3 IP