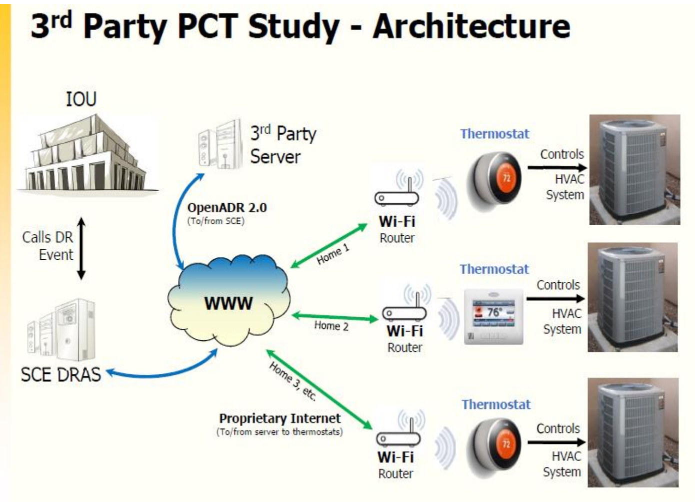
QualityLogic

OPENADR AND THE INTERNET OF THINGS

The classic Demand Response (DR) program has the utility, facilitator, or aggregator providing or partially funding the infrastructure required to shed load at a commercial, industrial, or residential site. This infrastructure may include communication equipment needed to receive DR event signals, telemetry usage measurement equipment, and configuration support for energy and building management systems. The costs to the entity managing the DR Program and the logistical effort required on the part of the entity that will shed load are significant barriers to the adoption of DR.

The emergence of cloud computing, high speed Internet, pervasive WiFi connectivity, and small low cost microcontrollers has enabled many manufactures to offer IP enabled, cloud connected consumer and business products. Some of these IP connected products, such as thermostats, have been leveraged by utilities to offer "bring your own thermostat" type DR programs. This leveraging of existing infrastructure to enable DR programs is significant in its potential to broaden the adoption of DR.

However, most of these programs require leveraging the thermostat manufacturer's cloud infrastructure to propagate the intent of the DR signal received via OpenADR to downstream thermostats over their proprietary networks. While there is nothing wrong with this approach, its extensibility is limited to manufacturers with the necessary cloud infrastructure and technical resources. A study conducted by Southern California Edison leveraging Nest and EnergyHub cloud connected thermostats yielded very encouraging results in terms of load shed participation and consumer perceptions of the program. See diagram below an from Edison presentation on this study.


3rd Party PCT Study - Architecture

The broad availability of low power, low cost wireless communication modules, microcontrollers, microcomputers, and systems on a chip enables what is commonly referred to as the Internet of Things (IoT). Thousands of products are leveraging IoT technology in every conceivable type of load consuming consumer product. These products can be remotely configured, controlled, monitored, and used in both simple and sophisticated home automation schemes. However, these products use a wide range of incompatible wireless physical layers, transport protocols, and device profile definitions. A few example variations include ZigBee, Z-Wave, Bluetooth, Insteon, and many others.   In spite of these challenges consumers are acquiring these technologies in increasing numbers.

This situation creates an interoperability nightmare for consumers who have a limited ability to understand what products will work together. An interim solution spearheaded by companies like Samsung, Staples, Home Depot, Lowes, and others is to create a central hub that can act as a coordinator and translator of products with these diverse communication methodologies. The software provided by the hub manufactures also provides the consumer with a single user interface for controlling all their home automation products. Popular hubs include SmartThings, Wink, Isis, Icontrol, and others.

Longer term solutions to this problem are standards based frameworks, such as OIC and Alljoyn, which enable discovering device capabilities and make the differences in communication technologies used somewhat transparent from an application layer programming perspective.

Today we have a sizable pool of consumers who own IoT enabled devices whose load profiles can be remotely altered through an IP connected hub in their home. These consumers are familiar with setting up rules that control the behavior of their connected home devices. All the necessary pieces are in place to implement DR in these connected consumer environments, with the exception of a simple way for the consumer to receive the standard OpenADR event signaling. This is not to say there are no products that enable this capability. Universal Devices, a contributor to the OpenADR standard, has a modestly priced home automation controller (ISY) that has OpenADR support. The ISY product is being used in a number of utility pilots.

The highest profile home automation hub on the market is Samsung's SmartThings hub, a technology acquired several years ago. Samsung has committed to roll out the SmartThings technology in a wide range of consumer electronics and appliance products. At the 2016 CES show, Samsung demonstrated SmartThings embedded in the user interface of their top-of-the-line high resolution TV offerings.

SmartThings has a web based open source development environment that provides support for creating "device handlers" for new IoT products and "SmartApps" for allowing consumers to control those products. The architecture is very elegant with mappings between physical device behaviors and a set of abstract capabilities. Any device that declares it is a "light" must support a number of common commands such as 'on', 'off', or 'dim'. Consumers can create simple scripts that can control any manufacturer's light bulbs that support the "light" capabilities.

QualityLogic took on a research project to see how difficult it would be to transform a $99 Samsung SmartThings hub into an OpenADR VEN that could receive DR signals and potentially leverage the entire hardware and software SmartThings ecosystem for load shed activities.

Our objective was to create a SmartThings device handler that played the role of an OpenADR VEN. When a DR event was received by the "VEN" device handler; the device handler would trigger predefined rules in a SmartThings SmartApp that would control the load shed activity.

To keep things simple, the VEN implementation had the following limitations:

- Pull only A profile VEN running over HTTP only
- Limits number of events in a payload from the VTN to 1
- Targets limited to the venID
- Will handle events signals with up to 3 intervals

Within these limitations, all the VEN behaviors are OpenADR profile compliant. For controlling load shed behavior we leveraged the large open source library of SmartApps, finding one called The Rules Engine. This easy to use smart phone application allows compound rules to be created for the entire ecosystem of available home control devices supported by SmartThings. For our project we installed a Cree ZigBee light bulb with dimming capabilities and a Fidure thermostat, also with a ZigBee interface. Using the Rules Engine, we created four rules that could be triggered by the receipt of messages from the VEN device handler. The rules were as follows:

- **Normal Usage Rule** - Cree Bulb dim at 100%, Thermostat in heat mode with setpoint at 70 degrees.
- **Moderate Load Shed** - Cree Bulb dim at 40%, Thermostat in heat mode with setpoint at 68 degrees.
- **High Load Shed** - Cree Bulb dim at 5%, Thermostat in heat mode with setpoint at 65 degrees.
- **Emergency Load Shed** - Cree Bulb off, Thermostat in off mode

For testing the SmartThing VEN, we used the QualityLogic VTN in the Cloud, based upon the EPRI open source VTN. The VTN was used to create a variety of events with SIMPLE signals using various interval values ranging from 0 to 3 corresponding to the rules defined in the Rules Engine.

While the implementation of the VEN and its testing with VTN in the Cloud was quite successful, the SmartThings development environment did pose a number of significant challenges:

- Code size was limited to 100K bytes
- Groovy/Java with a whitelisted set of standard classes
- No third party jars, no class definitions, single threaded execution
- Device handlers only execute upon receipt of trigger (device change, poll, events)
- Device handlers and SmartApps have a 40 second maximum execution time
- No disk storage, but does have persistent state properties
- Development environment with no sophisticated debugging environment (printf only!)

While it was a pain to deal with all these limitations during coding, it was not an insurmountable obstacle, and an OpenADR A profile VEN was successfully developed with only a modest effort.

The most significant obstacle encountered was finding a way to get the device handler to reliably poll the VTN. By default, the SmartThings infrastructure will trigger a device handler every 10 minutes if a "polling" capability is defined. However, past experience showed this default polling to be quite unreliable. We instead attempted to use a SmartApp called Pollster to call the device handler at one minute intervals using a SmartThings defined "schedule" method. However, this also proved not to be reliable as apparently this polling traffic is low priority and it gets dropped during times when the SmartThings infrastructure is busy. We finally resorted to using an Arduino Uno and SmartThings Shield to trigger periodic ZigBee device events in the VEN device handler to wake it up for execution. Not an ideal solution, but it worked reliably.

The code developed for this study could be OpenADR certified with a small amount of additional effort and validation that the SmartThings platform supports client certificate exchanges. While SmartThings is one of the only hubs to offer an open programming environment, this study demonstrates that with a minimal effort manufactures could OpenADR enable their hubs, opening up a large universe of potential load shedding resources to the DR community.