Building a Cable-Friendly Internet of Things

A Technical Paper prepared for SCTE•ISBE by

J. Clarke Stevens
Principal Architect, Emerging Technologies
Shaw Communications
1401 Lawrence St, Suite 1550
Denver, CO 80202
720-723-2316
clarke.stevens@sjrb.ca

Table of Contents

List of Figures

Introduction

The Internet of Things (IoT) is driving new products and technologies—as well as customer adoption rates. This increased integration presents an exciting opportunity for cable operators to reinforce their value, and support customers as they transition to an increasingly interconnected home. With this opportunity comes a handful of challenges around network security, device interoperability and increased customer support requirements for installation and troubleshooting. The *Open Connectivity Foundation* (a consortium of CableLabs, Comcast, Shaw, Cox, Midco, Mediacom and over 400 other companies like LG, Honeywell and Cisco) has been formed to overcome these challenges through a standardized approach to IoT.

The benefits of cross-organizational collaboration include industry alignment, cost-savings and an opportunity to develop mutually-beneficial products and platforms. For the cable industry, this partnership promises a seat at the table with leading device manufacturers, as well as economies of scale for troubleshooting and alignment on security, interoperability and scalable remote manageability. The partnership's work to develop an ecosystem which considers these impacts and allows for seamless integration of new devices will ultimately benefit customers and help usher in a new era of interconnectivity.

Executive Summary

In this paper, we explore some of the major challenges faced by cable operators implementing an Internet of Things (IoT) offering for their customers. The IoT exponentially increases in number of connected devices on MSO networks. At the same time, these connected devices act more directly on the real world – controlling expensive equipment, unlocking doors and otherwise connecting the real world to the online world. This new type of network imposes an increased operational load on the operators while introducing new risks for both customers and the operator's network.

A cable-friendly IoT will make this workload manageable and reduce the risk. OCF is explored as a potentially important system that addresses these issues directly because cable operator requirements were part of the original design of the system. These requirements have been successfully implemented by OCF. OCF has also created a very complete development system that allows devices to be created automatically and very quickly. An IoT device based on OCF will be built live on stage during the presentation.

1. Context

When it comes to the Internet of Things, it is such a vast and varied topic that any particular discussion requires a bit of context. This paper will consider the IoT from a perspective of individual devices (lights, thermostats, etc.) and the applications that control them. The main topics to be discussed are the technical aspects of making these devices secure, getting them to communicate regardless of manufacturer or networking technology, and making things simple for both developers building the devices and the ordinarly lay customers using the devices. In this context, operators are providing a valuable service to their customers in a way that can be profitable. This paper does not consider derivative services that can be based on the data associated with an IoT offering.

The following diagram provides an overview of the physical architecture of a comprehensive IoT network. The IoT service rides on top of existing operator data services, but should be considered as a separate platform for an unlimited number of specific IoT products to different customer segments.

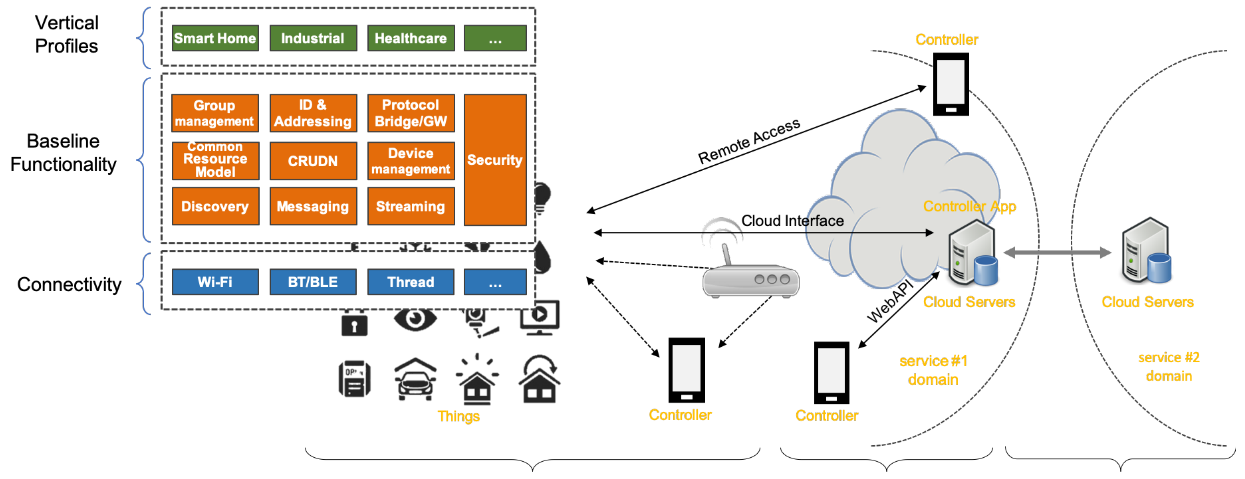


Figure 1 – A connectivity view of the Internet of Things

2. Operator Concerns

Operators find themselves in a unique position concerning the IoT. Cable operators are ideally qualified to be providers of the Internet of Things. They have fast networks. They already manage subscription services for customers. They have a strong 24/7 connected support system as well as fleets of technicians who can resolve problems both on premise in person. Additionally, they have relationships with equipment vendors and are accustomed to installing complex devices in customer homes and places of business.

The Internet of Things, however, comes with a host of challenges. The potential number of connected devices in a local network could be in the hundreds. The customer wants to buy the products that meets their needs regardless of manufacturer or what the operator wants to support. Each of these devices poses management challenges, enables a new attack surface for network criminals and could even threaten the integrity of the entire operator network. As with traditional cable video business, there is a great opportunity here for cable operators to be aggregators who simplify the customer experience and offer reliable, in-demand services at a great value. In order to do this effectively, the operator needs to provide the best network security available. At the same time, there must be a rich source of products that operators can provide – and support – to give customers the technology they want in a safe and easily-manageable environment.

2.1. Security

Security cannot be an afterthought. Security must be integrated into IoT systems from the planning phase to ensure that there are no obvious security gaps. Systems should be designed to incorporate the newest security principles, anticipate potential security breaches and have a plan for addressing them.

- **Devices** – Devices are the on the front lines of potential attack so they must implement security best practices. This includes best-in-class security algorithms, the ability to update firmware on the device, and the ability to quarantine devices that can't be controlled.
- **Network** – The network can be defined as the collection of devices connected to it and the actual connection between those devices. Securing individual devices is a start, but the paths beyond the local network are also at risk. Malicious devices can be blocked to prevent them from creating havoc on the broader network. Tools that enable careful monitoring of network traffic can indicate sources of attack and isolate them before they cause too much trouble.

- **Privacy** – Privacy is related to general security but is concerned with protecting the identity of network customers. Most customers have little understanding of, and no feasible defenses to protect their privacy, so it is incumbent on network providers to do this for them.

2.2. Interoperability

Interoperability gives customers the freedom to install devices with features and services they want while allowing them to choose from a large pool of products from different vendors. In the current IoT environment, there are numerous smart phone applications that only support devices from the same company. This is untennable. Some vendors have seen this problem and have used the opportunity to provide interfaces to these products individually to enable a single controller. This is an expensive interoperability solution. A more efficient solution would provide a common framework compatible with many devices.

Devices – To provide scalable interoperability, devices need an open interface that complies with a standard in order to drive scalable interoperability. The benefits of standards are evident all around us. Web sites, mobile phones and electrical sockets are all good examples of the potential of standards. If there is only one standard, everything works together. Where there are several (as with electrical sockets worldwide), you need adapters. The fewer adapters you need, the better.

Controller Applications – Controller appliations are the flip-side of devices. If devices are based on different standards, the controller must implement each of them, along with translators, in order to enable any level of interoperability. This problem grows exponentially with the increase of devices on the customer network.

Groups, Scenes and Rules – Groups allow devices to be controlled in tandem. For example, a group of lights can be turned on together or dimmed at the same rate. Scenes allow a favorite group of settings to be remembered and reproduced later. Rules monitor various conditions (time, weather, door sensors, etc.) to be monitored. When the right conditions exist, further actions can be taken (for example, sounding an alarm or turning on a light). Interoperability allows different devices to work as groups to implement scenes or launch other actions.

2.3. Simple Development and Support

Most operators have little interest in producing IoT products themselves. Instead, they are interested in developing relationships with a few key partners or enabling a few key ecosystems. Customers, however, do not want to be limited on which devices will work with their IoT network. The market will provide these devices, but it becomes prohibitively expensive to support multiple ecosystems on a device or write new code from scratch.

Vendor Development – The Internet of Things relies on connecting common devices to the Internet. These products become harder to sell if they are much more expensive then their non-connected equivalents. So the IoT part needs to be inexpensive. This can’t be accomplished if you have to include software to support multiple ecosystems. Duplicating device data models, or supporting multiple ecosystems quickly becomes expensive. This is especially true for devices that have minimal memory or processing power.

Operator Development – Operators need to enable good customer experiences. The sheer number of IoT devices makes it impractical to deploy trucks and technicians to install every new device. This problem can be mitigated by making self-install extremely simple. If the customer does need help, customer service people need software tools to get an accurate view of the network and the ability to fix problem.

As with the IoT products themselves, the operator can scale support only if the interface to support devices is simple and consistent.

Freedom to Select Devices – One of the advantages of supporting an interoperable environment is that operators can choose which devices to support. Customers will ultimately decide which devices they want. If the operator already supports those devices, the customer is more likely to choose the operator-provided interface.

Universal Backend Architecture – In order to really make the IoT work operationally, devices, data and management need to rely on some common tools. It's hard to make a case for many IoT services independently. Devices run on different systems, connected through different networks. Still, they need common management, a uniform approach to data and the ability to control their security.

Network Independence – There are many different types of networks - Ethernet, Wi-Fi, Bluetooth, ZigBee and several mobile networks – and each has its individual strengths and weaknesses. Those strengths and weaknesses serve useful purposes, so it doesn't make sense to try to move everything to a common network. Instead, the control, manangement and data features that are independent of the particular network should be used to provide a common infrastructure.

Many IoT ecosystems are less about basic connectivity and more about the data produced by the individual devices. The principles behind data collection, however, are common. What is the format of the data? How often should it be sampled, or is it spontaneously generated? Where should it be stored? Should it be analyzed before it is collected or should it be retrieved in a raw state? These questions are independent of the particular ecosystem being used or how the device is connected. This fact makes data a good candidate for normalization in an IoT system.

Common Support Infrastructure – Regardless of the number of systems installed to minimize customer calls, customers are going to call. In order to efficiently dispatch these calls, the customer service representative must have tools that allow him to interact with the customer's network and IoT devices. This support will shorten customer calls and improve customer satisfaction. This type of tool requires a system where devices are recognized and controlled in a common way. They must also be managed from a security perspective in a way that is comfortable for the customer to allow and simple for the customer to enable and disable.

2.4. Other

There are a few other aspects of an IoT offering that might be considered by an operator in order to create a more universal and stable service.

International Standard – It helps if an IoT product can be based on international standards. A standard will include explicit instructions on how a device should be designed, constructed and how it should communicate. A standard has a greater chance of adoption as it is more likely to be supported by other companies in the ecosystem. Also, standards are likely to be maintained for some period of time by the authoring bodies by which they were produced. If the standard is open, it is easier for others to get access and to possibly influence how it works. This brings in more voices and more people who can review a standard to make sure it is consistent, efficient and implementable.

Open Source Implementation – While a standard goes a long way to enabling interoperability, there is still a chance that it will be interpreted differently by different readers. Despite the most valiant efforts of implementers, different interpretations are likely to lead to incompatible devices. This possibility can be reduced by an open source implementation. Creating a valid working example helps to clarify the specification in the standard and is likely to be used by others even if the specification is not clear enough.

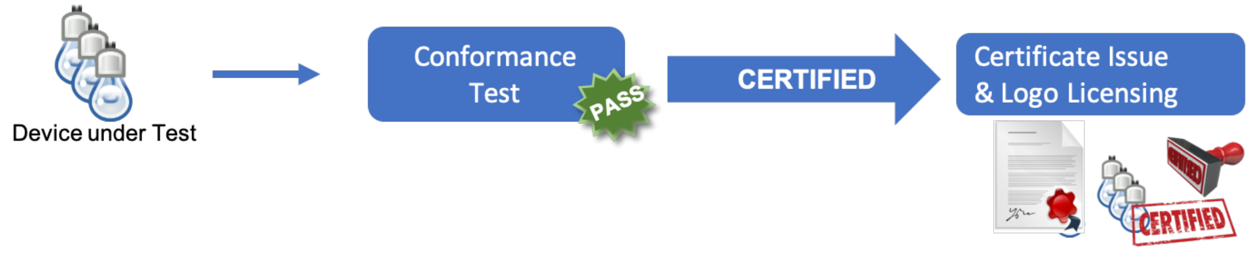


Figure 2 – The certification process. An automated test tool usually verifies compliance

Certification – The final check on an interoperable implementation is a certification process. A certification tool is really a reciprocal implementation with clear tests for each of the specified requirements. Of course the standard must be written with sufficient requirements to ensure interoperability and those requirements must be testable. Since some requirments are not testable by their very nature (negative requirments are sometimes an example of this), a certification tool can be augmented with attestation statements that an implementation is compliant with untestable requirements.

3. Open Connectivity Foundation Features

All the objectives stated above are not met by any single system at this point. Indeed the variability of these objectives between different operators indicates that there will always be a different combination of solutions for each operator. However, many of the objectives can be met by the Open Connectivity Foundation (OCF).

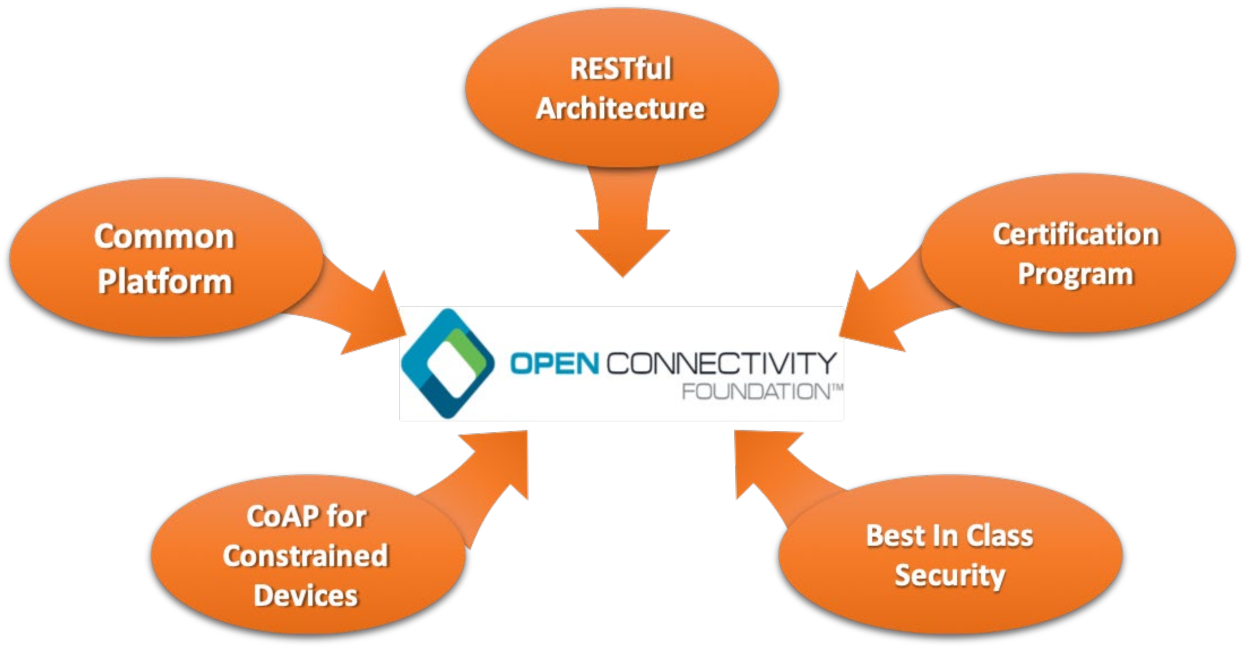


Figure 3 – Features provided in the OCF Internet of Things

3.1. Organization and Authorization

The Open Connectivity Foundation is an international standards development organization that writes standards for the IoT.

In OCF, there is something called three-pillar alignment. The standard, an open-source implementation (called IoTivity) and an automated tool are synchronized about every six months.

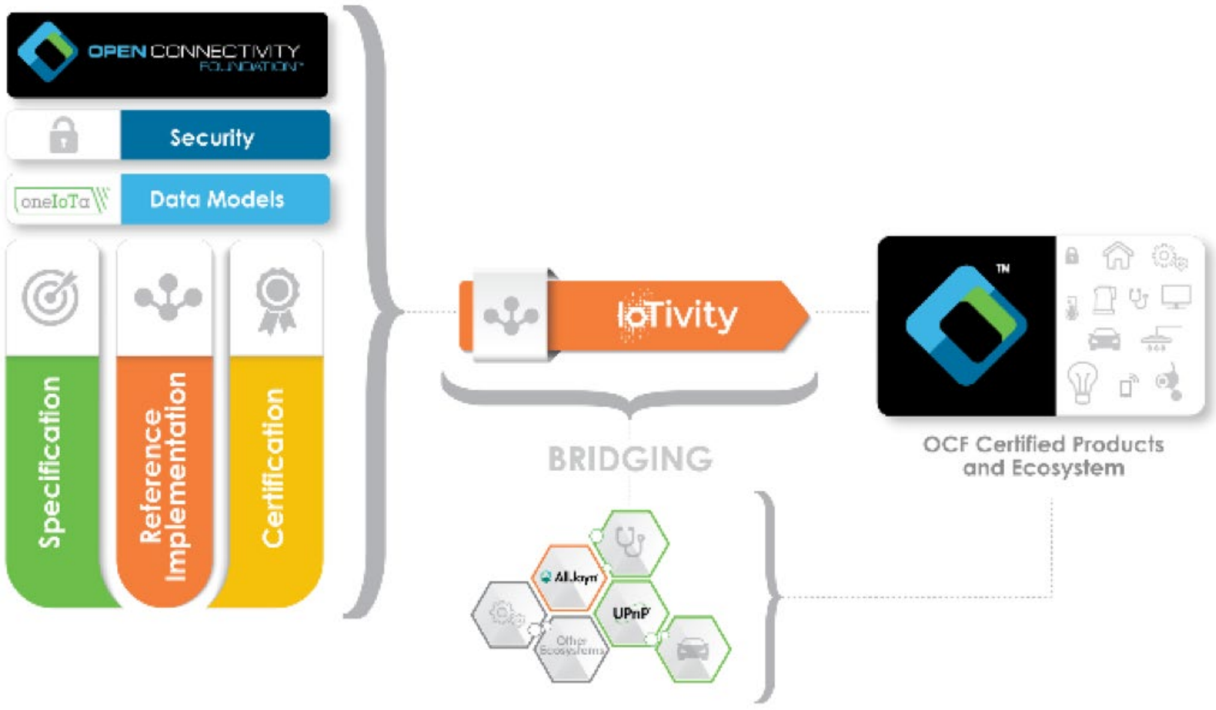


Figure 4 – OCF architecture featuring three pillar alignment (specification, open-source reference implementation and certification of devices)

The standard is frozen at a particular version number, then all contributors are notified in case they want to claim any intellectual property. This frozen version is the reference for the open-source implementation and the basis for the certification test tool (CTT).

The open-souce implementation must implement all requirements specified in the standard at each release version. The open source implementation may also include optional features or features that are not even specified, but at a minimum, it must implement all mandatory features. Compliance is validated with the automated Certification Test Tool.

The automated Certification Test Tool is a software program that implements all the test cases noted by SHALL statements in the standard. Each test case will instruct the device under test (DUT) to set up the particular conditions of the test case, then verify that the DUT responds as specified. In three pillar alignment, the open source implementation must pass all the tests for a particular specification version number.

The success of any standard is determined by how broadly it is adopted. One way to encourage adoption is through membership. Those companies willing to join the organizationand put in the work to develop the standard, the open source implementation or the test tool are quite likely to also implement the standard in their products.

In addition to memberships, liaisons are a valuable means to encourage adoption or at least cooperation. Liaisons are particularly valuable for OCF. Since the primary objective of OCF is to create a secure interoperable system for IoT regardless of the underlying protocols, the success of OCF is partly dependent on bridging to other ecosystems. This is done on the physical level through hardware bridges with interoperable data translations, but the liaison relationship aligns the cooperation on a business level.

3.2. Security

Security is arguably the most important aspect of the IoT. Since IoT devices manipulate real-world objects under networked electronic control, there is a serious risk of doing something like opening a door lock without proper authorization. State-of-the-art security is the most practical solution.

3.2.1. *Authentication*

Authentication is the process of verifying that a network, a device, or a user is who or what they claim to be. This is done through digital signatures and an infrastructure that can be trusted. OCF has set up multiple registration authorities to generate and hold the root of trust for OCF certified devices. Certificate authorities then are then used to allow devices to follow the path to the root of trust to verify the authenticity of OCF devices.

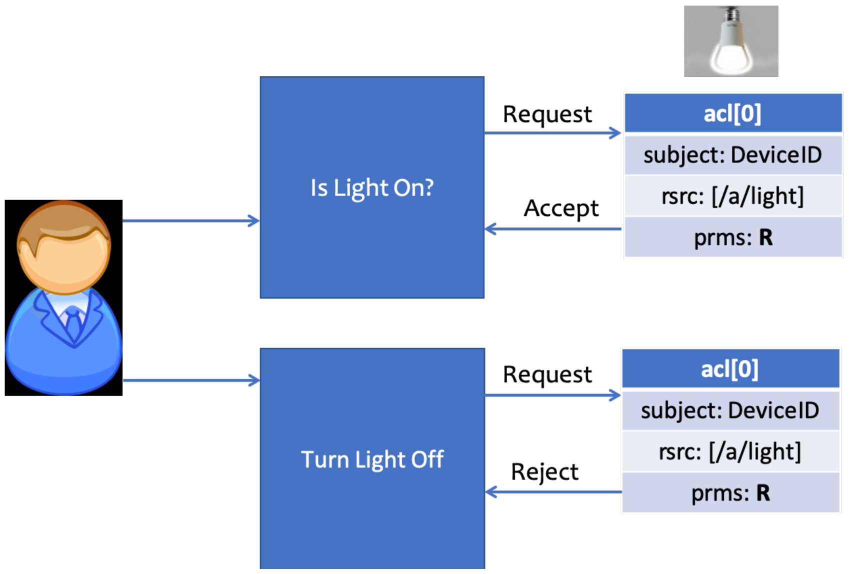


Figure 5 – A read-only device (R) cannot be controlled unless an authenticated user obtains authorization

3.2.2. *Authorization*

Once a device is authenticated and its ownership status is verified, it can be trusted to be onboarded (joined to the secure OCF network) and controlled by authorized users.

- **Individuals** – Authorization in OCF is managed by the "owner" of the OCF secure network (a subset of the the user's network). In a normal customer network, the owner is assumed to be the first user to onboard a device. The owner of the OCF network is authorized to authorize other users. The permissions on a particular device can be given to a particular authorized user.
- **Roles** – Roles allow for simpler management of authorization by assigning any user who has been assigned a particular role to share the authorizations of that role. Common roles include administrator, user and guest. Other roles can be created as needed.

3.2.3. Security Profiles

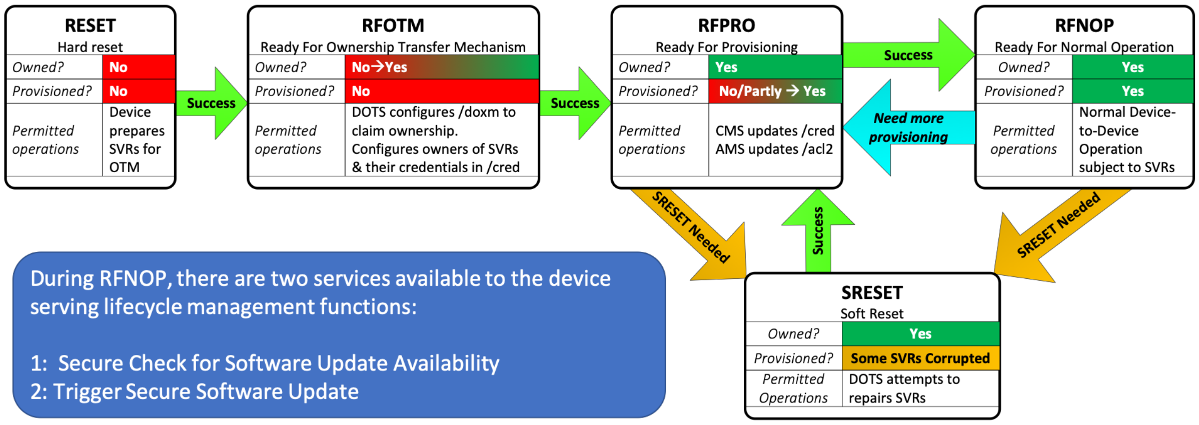
Security profiles define a minimum level of security for all OCF devices. OCF supports both symmetric and asymmetric security methods, but every certified OCF will have a minimum level of security.

- **Just Works** – Just Works security is the simplest form of security allowed in OCF. It is based on Diffie-Hellman symmetric keys. The public keys are distributed in advance and the secure connection is set up pairwise between a client and a server. This fact limits the scalablilty of Just Works security as it requires N x N security relationships to be set up.
- **Random PIN** – Random PIN security also uses Diffie-Hellman, but relies on a random PIN generated by the server during onboarding. This strategy makes it harder for someone without physical access to the server screen to try to steal the connection during setup.
- **Certificates** – The other three security profiles are based on certificates issued by a Certificate Authority (CA). These certificates, in turn, are based on a root-of-trust issued by the Registration Authority under the management of a Management Authority and using policies provided by the Policy Authority. The OCF board of directors is the Policy Authority. The other authorities are contracted out. There are currently three certificate-based profiles. There is not a security hierarchy. Rather each profile has different features that can be useful in different situations
  - **Black** – The black profile uses public key infrastructure (PKI) defined by OCF and signed by designated OCF authorized certificate authorities. This guarantees that security meets the requirements defined in the OCF certificate polic.
  - **Blue** – With the blue profile, manufacturers are allowed to specify their own certificate authorities. They certify that they conform to the OCF-defined security criteria.
  - **Purple** – This profile supporte a requirement to a piecewise boot process and secure online software update. This improves device integrity.

3.2.4. Onboarding

Onboarding is the process of connecting a device to the secure OCF network. The secure OCF network is really just a collection of devices that will only communicate with other similarly secure devices. In OCF, this secure "network" sits on top of the IP network. So a device must first be connected to an IP network (via Ethernet, WiFi, bridging, etc.) before the OCF onboarding process.

The onboarding process, then, is primarily using one of the security profiles described above to bring an authorized device on to the network and connecting it with authorized client devices.


RESET
Hard reset
Owned? No
Provisioned? No
Permitted operations: Device prepares SVRs for OTM
Success
RFOTM
Ready For Ownership Transfer Mechanism
Owned? No→Yes
Provisioned? No
Permitted operations: DOTS configures /doxm to claim ownership. Configures owners of SVRs & their credentials in /cred
Success
RFPRO
Ready For Provisioning
Owned? Yes
Provisioned? No/Partly → Yes
Permitted operations: CMS updates /cred AMS updates /acl2
Success
Need more provisioning
RFNOP
Ready For Normal Operation
Owned? Yes
Provisioned? Yes
Permitted operations: Normal Device-to-Device Operation subject to SVRs
SRESET Needed
Success
SRESET Needed
SRESET
Soft Reset
Owned? Yes
Provisioned? Some SVRs Corrupted
Permitted Operations: DOTS attempts to repairs SVRs
During RFNOP, there are two services available to the device serving lifecycle management functions:
1: Secure Check for Software Update Availability
2: Trigger Secure Software Update

*Device can transition to **RESET** from any state (these transitions are not shown)*

Figure 6 – The process for securely connecting an OCF device to the secure OCF network (which is implemented over a standard Internet Protocol (IP) network

3.2.5. *Secure Sharing*

3.3. Interoperability

The second objective for OCF is interoperability. There are a number of features of the OCF architecture that make interoperability simple to implement. These features include a common data model, a RESTful architecture, cloud support and bridging.

3.3.1. *Data Models*

Data models are at the center of OCF. Data models describe resources that are composed into complete devices. These data models are used to generate some specifications, fully functional source code and test scripts.

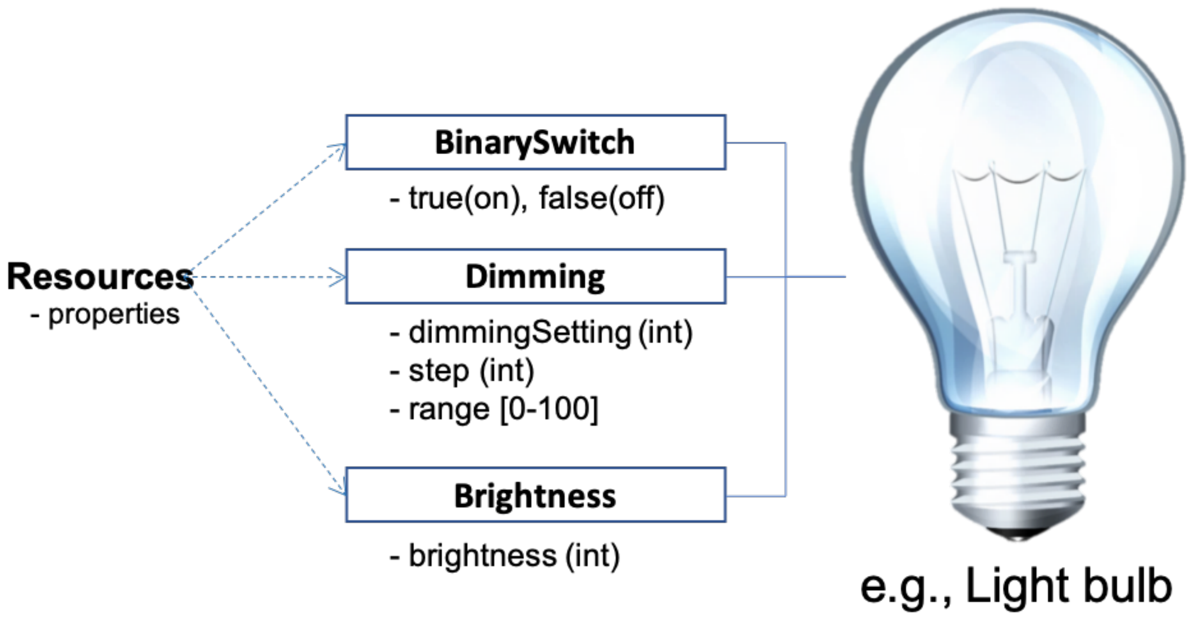


Figure 7 – The definition of a dimmable light bulb using component resources

Core Models – The core data models are part of the core architecture in OCF. Since these models have aspects tied to the OCF architecture and particular protocols they are kept separate from the resource and device models.

Resource Models – Resource models are models that contain the minimum properties and definitions to describe a complete component. For example, a temperature. Interoperability in OCF is defined at the resource level. This allows for interoperability between any devices that use a common resource.

Device Models – Devices are fully functional items that a consumer might purchase. Devices are composed of a collection of resources. Device models are defined with the minium number of required resources to be classified as a certain type of device (e.g. a thermostat). By defining devices in this way, a manufacturer is able to arbitrarily enhance their device to distinguish it from competitor devices while still be compatible with other such devices.

Resource models are managed within the oneIoTa online tool. This tool includes a simplified Integrated Development Environment (IDE), and process management to enable the review and approval of new resource data models. Once models are approved, they are pushed to a git repository where they can be obtained to build OCF devices.

Other Organization Modesl – The oneIoTa tool supports models contributied by other organizations. Any organization can set up an independent organization within oneIoTa to use both the IDE and the process management features. This allows other organizations to independently control their own models while making them accessible to OCF.

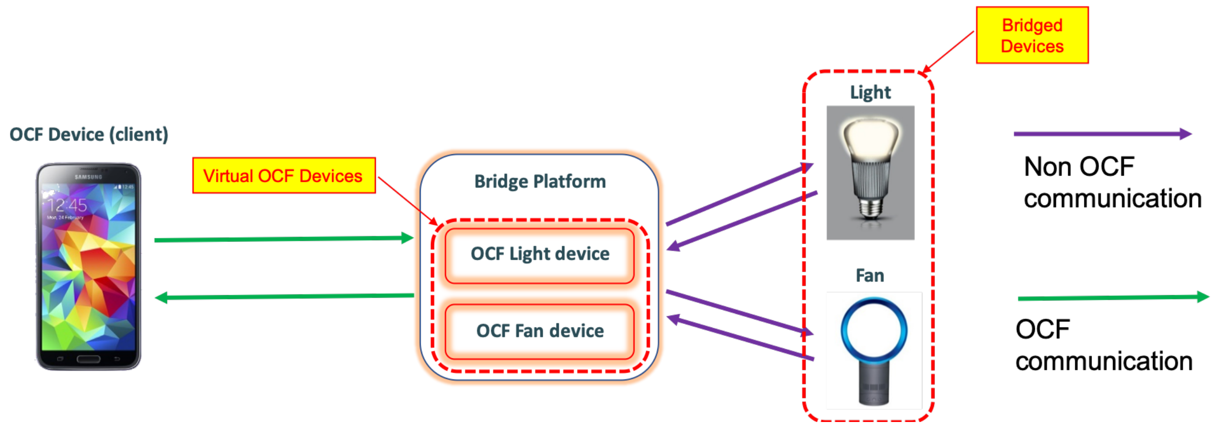


Figure 8 – An asymmetric OCF bridge between an OCF client and non-OCF server devices on a non-OCF network

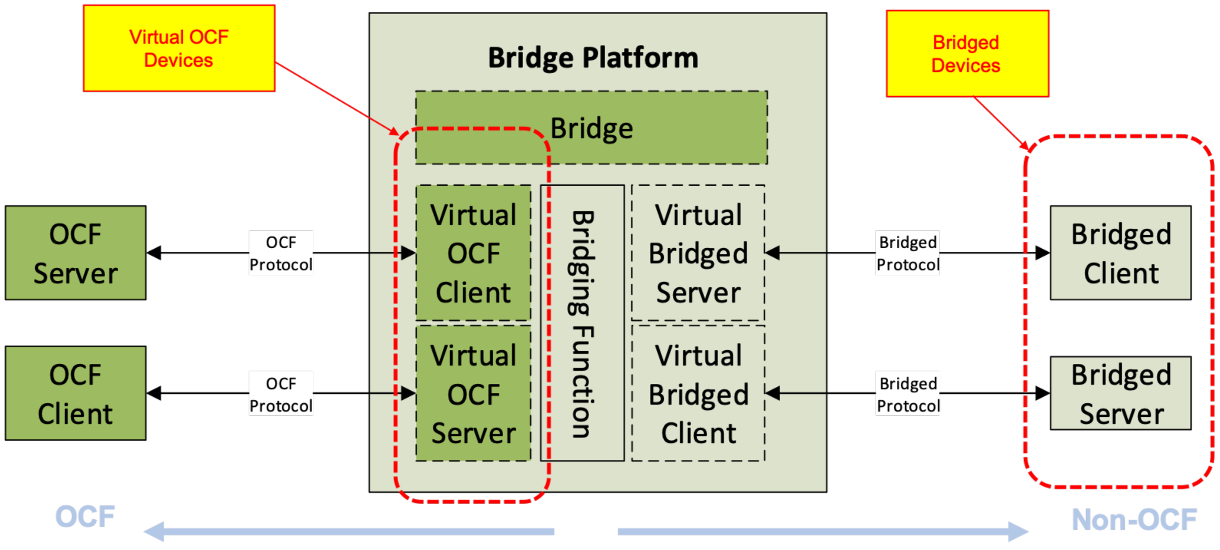
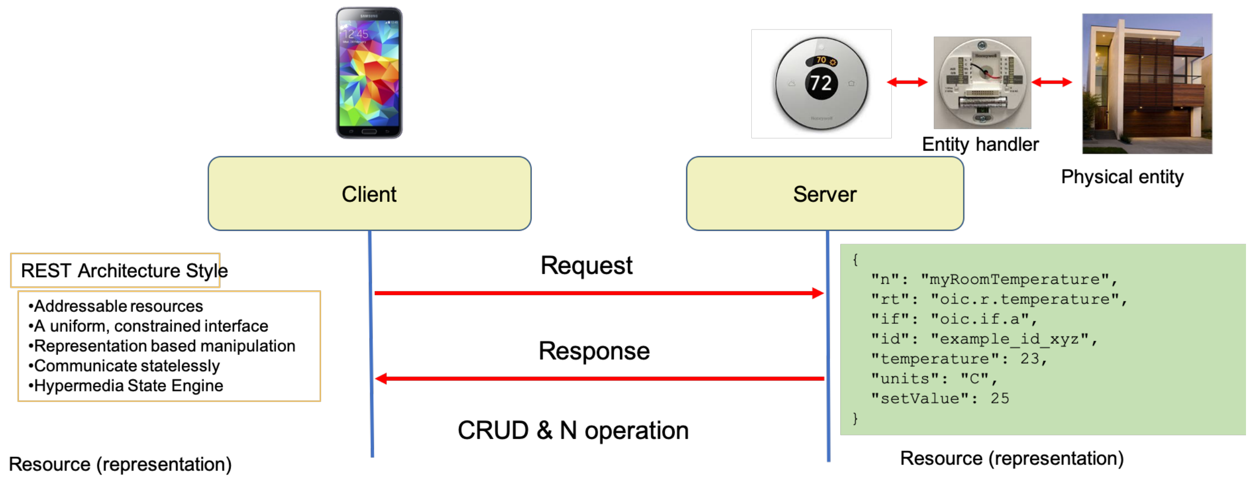


Figure 9 – A symmetric OCF bridge where client and server devices may be in either network (i.e. OCF clients can control non-OCF servers and non-OCF clients can control OCF servers)

Derived Models – Derived models describe the mapping of data between native OCF models and the models from any other organization. The mapping describes mapping into and out of OCF. The mappings do not need to be one-to-one. In fact, any procedures that can be done in most programming language can be used to map between the models. This flexibility makes it possible to build bridges between OCF and most any other ecosystem.

3.3.2. *Restful Architecture*

OCF supports a RESTful architecture. Devices are normally controlled using Create, Retrieve, Update, Delete, and Notify with data models. However, OCF can also be used to remotely execute commands. This is useful for supporting streaming services and other services that are not well-modeled using a RESTful approach.



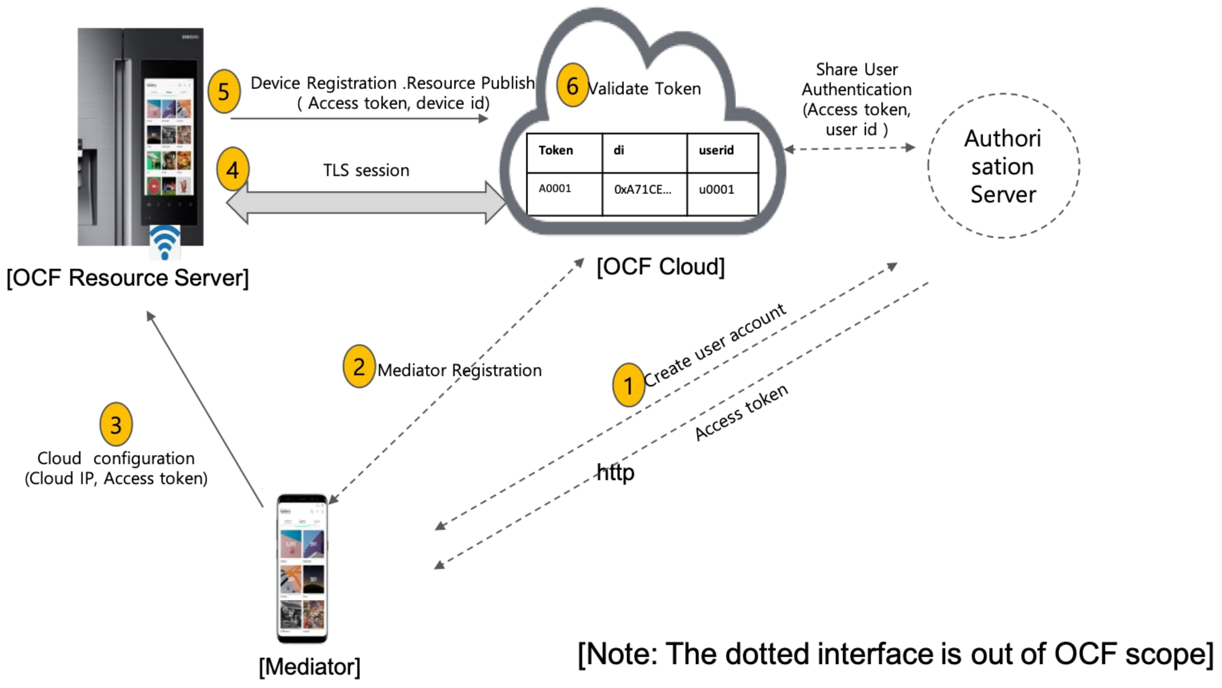
```
{
  "n": "myRoomTemperature",
  "rt": "oic.r.temperature",
  "if": "oic.if.a",
  "id": "example_id_xyz",
  "temperature": 23,
  "units": "C",
  "setValue": 25
}
```

Figure 10 – A RESTful system in which any client can make requests to any server at any time. Notification is also supported, so servers can report events to be handled by clients

3.3.3. Remote Access and Cloud Support

The IoT is not truly an Internet experience if it is only on a Local Area Network (LAN). For this reason, OCF supports remote access to LAN IoT devices (e.g. using a smart phone) as well as cloud services and cloud-to-cloud connections between vendors.

Device-to-Cloud – Device-to-cloud supports connecting servers or clients to devices in the cloud. This means not only that devices can be controlled from the network but that virtual devices can be implemented in the cloud.


5 Device Registration .Resource Publish (Access token, device id)
4 TLS session
6 Validate Token
Share User Authentication (Access token, user id)
Authorisation Server
[OCF Resource Server]
[OCF Cloud]
2 Mediator Registration
1 Create user account
Access token
http
3 Cloud configuration (Cloud IP, Access token)
[Mediator]
[Note: The dotted interface is out of OCF scope]

Token	di	userid
A0001	0xA71CE...	u0001

Figure 11 – The architecture and operation of the OCF cloud

Cloud-to-Cloud – Cloud-to-cloud allows clouds from two different companies or ecosystems to make connections between devices and clients anywhere. This means companies can define their own cloud infrastructures and still use OCF and bilateral deals to control OCF devices regardless of particular network connectivity.

3.3.4. *Bridging to Other Ecosystems*

Bridges in OCF allow connections between OCF and other ecosystems. Bridges connect between protocols at layers beneath OCF, then use OCF derived models to define the mapping between OCF and other ecosystems. This approach has the added benefit of providing compatibility with devices that have already been deployed as long as an OCF bridge is added to the system.

3.3.5. *Common Management*

OCF enables connectivity between devices from different ecosystems through OCF bridges or cloud connections. This strategy enables OCF management tools to extend to these other ecosystems. Tools that allow for devices to be onboarded, controlled and updated can all be used with only an OCF interface. This means there are lower costs for deployment and ongoing management of mixed IoT ecosystems.

Figure 12 – OCF developer event in China

3.3.6. *Interop Events*

Interop events hosted be OCF give vendors the opportunity to test their servers and clients in a safe setting with servers and clients from other vendors. This demonstrates the OCF promise of security and interoperability. It also gives manufacturers confidence that their devices with be interoperable when deployed to customers.

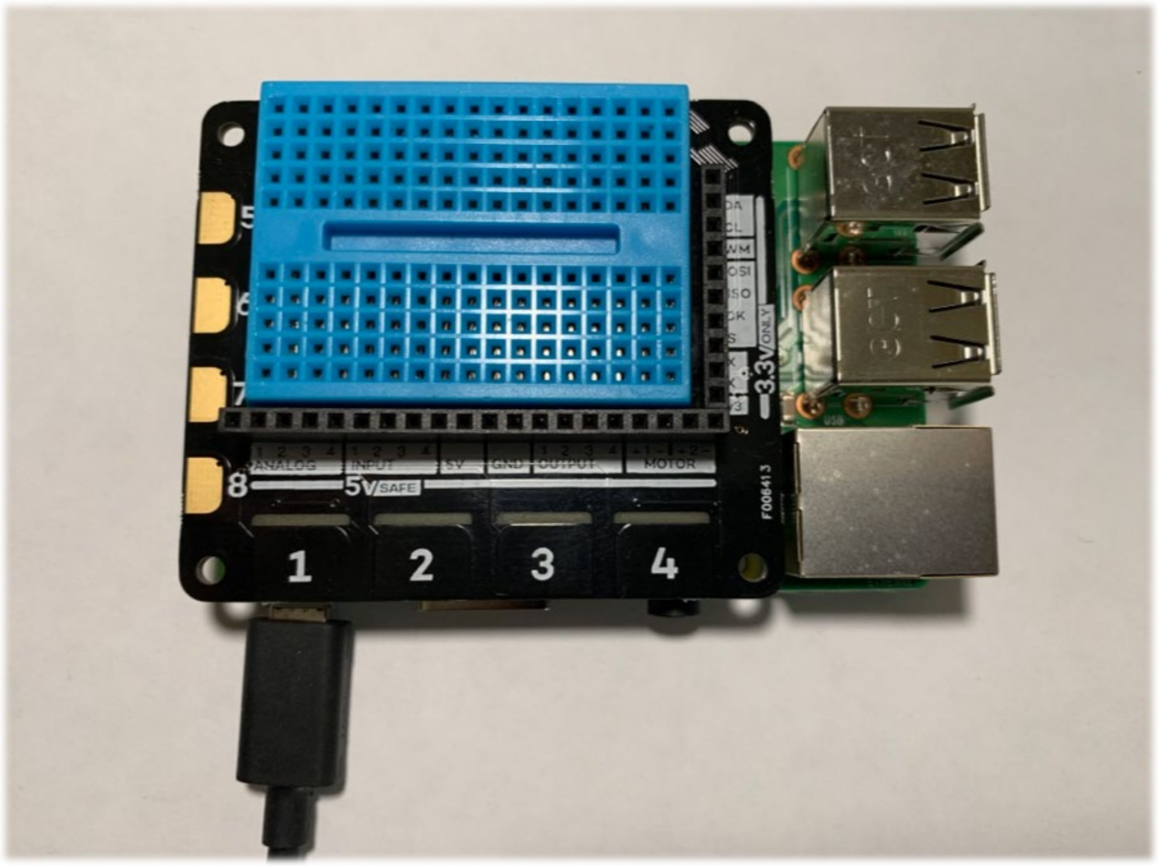


Figure 13 – The Raspberry Pi board used in the tutorial (with Pimoroni Explorer Hat Pro daughter board)

3.4. Simple Development

The final objective of OCF is to make it easy for vendors to create, deploy and manage devices. OCF has several tools that make this easy.

3.4.1. Simple Device Description

A device is simple to define in OCF using JSON schema. Each device is described as a collection of the individual resources it implements. A thermostat, for example, would have a setting for the desired temperature, a thermometer for measureing the current temperature, and a switch that can be used to turn on the heating or air conditioning. In OCF, these descriptions are written in JSON (a data modeling language). An example is shown below:

```
[
  {
   "path" : "/binaryswitch",
   "rt"   : [ "oic.r.switch.binary" ],
   "if"   : ["oic.if.a", "oic.if.baseline" ],
   "remove_properties" : [ "range", "step" , "id", "precision" ]
  },
  {
   "path" : "/oic/p",
   "rt"   : [ "oic.wk.p" ],
   "if"   : ["oic.if.baseline", "oic.if.r" ],
   "remove_properties" : [ "n", "range", "value", "step", "precision", "vid"  ]
  }
]
```

Figure 14 – Example input file for a simple binary switch

3.4.2. Code Generation

OCF has a tool called DeviceBuilder that will generate working OCF code using the device description file described above, a common code template (C and C++ developed so far), and the resource descriptions stored in oneIoTa.The generated code will compile without changes to create a working OCF IoT device. The code includes stub routines that can be populated with interface code to the hardware of a specific real-world device. Usually this is only a few lines of code that is linked to a library.

3.4.3. Building the Application

Building the application is simply a matter of compiling and linking the code that was generated by DeviceBuilder. A makefile is used to describe all the dependencies and include the appropriate hardware support libraries.

3.4.4. Device Ownership

The lowest level device security model supported is called “just-works” and is based on Diffie-Helman symmetric security. A variation can also be implemented with a random PIN being generated by the server device and entered in the client device. Asymmetric certificate-based security is also supported in OCF.

3.4.5. Introspection and Automatic User Interface Generation

In addition to code generation, the DeviceBuilder script creates an “introspection” file that is used to create the user interface for the newly created device. The client applications (OTGC) reads this file and creates a generic user interfaces (genereally buttons and text) that can be used to control any OCF server device. OTGC is available for Android, iOS, Windows and Linux and the source code is available, so vendors can modify it to add their own widgets for a nicer looking interface.

3.4.6. Running, Testing and Debugging

The device can be run by resetting the security of the device to Ready For Onboarding Transfer Method (RFOTM), the using the run script. Besides using OTGC, the developer can also run Device Spy to have explicit control of the exact message payloads that are sent.

4. OCF Development Process

Here is an example of the entire build process:

1. Start with the following device description file:

```
{
  {
    "path" : "/touch1",
    "rt"   : [ "oic.r.sensor.touch" ],
    "if"   : ["oic.if.baseline", "oic.if.a" ],
    "remove_properties" : [ "range", "step" , "id", "precision" ],
    "remove_methods" : ["post"]
  },
.
.
.
  {
    "path" : "/output4",
    "rt"   : [ "oic.r.switch.binary" ],
    "if"   : ["oic.if.baseline", "oic.if.a" ],
    "remove_properties" : [ "range", "step" , "id", "precision" ]
  },
  {
    "path" : "/oic/p",
    "rt"   : [ "oic.wk.p" ],
    "if"   : ["oic.if.baseline", "oic.if.r" ],
    "remove_properties" : [ "n", "range", "value", "step", "precision", "vid"  ]
  }
}
```

Figure 15 – Partial input file for the tutorial

2. Now generate the source code from the device description file by using the DeviceBuilder script:

   gen.sh

3. Now that the source code has been generated, compile and link it:

   build.sh

4. The server is now created. Next, set the security mode to RFOTM:

   reset.sh

5. Finally, run the server code:

run.sh

6. With the server running, run the OTGC (using Android) client.

7. Press the "discovery" button to find the server.

8. Now select the discovered server and press the onboarding (+) button.

9. With the server onboarded, launch the automatically-generated user interface.

10. Test the user interface by moving the binary switch. The effect will be reflected in the server window.

4.1. Controlling the Server with DeviceSpy

DeviceSpy is a lower-level client that gives developers explicit control of the OCF payload. It implements the same functionality as OTGC, but is available only as an executable application on Windows.

4.2. Testing with the Compliance Test Tool

The final step of the development process is to test the device for compliance to the OCF specifications. This is done with a tool called Compliance Test Tool (CTT). This automated tool will test almost every aspect of an OCF implementation. There are a handful of requirements that cannot be tested by CTT. For certification, an implementation must pass the CTT and also attest to compliance with any of the OCF requirements are not testable.

Conclusion

As mentioned at the beginning of this paper, the cable industry is well-positioned to become the premium IoT provider. While the industry has several natural advantages in the IoT space, there are serious security challenges it will need to address. Additionally, it has been difficult to find any single applications or services that justify the significant investment in infrastructure required to support an IoT service. That means the investment must support several different business cases on the same platform. Also, the Internet of Things exponentially increases the number and complexity of Internet-connected devices on the network. These devices must be easy to build, manage and support.

Open Connectivity Foundation makes up a piece of this puzzle. While its scope does not cover all cable industry IoT use cases, it does provide best-in-class security, a comprehensive infrastructure for interoperability and tools to support quick development.

Cable operators still have many problems to solve in this area. There are many industry sectors that are not adequately addressed by OCF. Even the sectors that are well-covered are just beginning to be deployed. The ability to be managed is not the same as having a management infrastructure. Still, it is a start. OCF can be used to as a first example of an open, secure, interoperable platform that can be used in key cable IoT markets.

Abbreviations

OCF	Open Connectivity Foundation – an international standards development organization for the Internet of Things
IoT	Internet of Things – the collection of objects that interact with the real world and are enhanced by connection to the Internet

Bibliography & References

openconnectivity.org, Open Connectivity Foundation, https://openconnectivity.org, 2019.

iotivity.org, IoTivity: Getting Started, https://iotivity.org/getting-started, 2019.

iso.org, ISO/IEC 30118-1:2018, INFORMATION TECHNOLOGY — OPEN CONNECTIVITY FOUNDATION (OCF) SPECIFICATION, https://www.iso.org/standard/53238.html, 2018.