Principles for Interoperability in the Internet of Things

A Technical Paper prepared for SCTE/ISBE by

J. Clarke Stevens
Principal Architect, Emerging Technologies
Shaw Communications
2420 17th Street
Denver, CO 80202
587-393-0605
clarke.stevens@sjrb.ca

Table of Contents

List of Figures

List of Tables

Introduction

Perhaps the biggest problem with the emerging Internet of Things (IoT) is that there are so many standards and proprietary systems. Apple, Google and Amazon each have proprietary approaches. Standards on relevant IoT topics are available from IEEE, IETF, W3C, ISO/IEC and virtually every other standards body. Even organizations trying to comply with standards are forced to choose between incompatible options. The integrated Smart Home and other IoT applications and systems cannot achieve their promise if the billions of connected things can only connect with a limited subset of the other devices. This paper will outline a number of principles that can enable interoperability, scalability and security while including all of the devices that need to be connected. The Open Connectivity Foundation (OCF), as one of the premier standards organizations trying to solve the interoperability problem, will be measured against these principles. The SCTE IoT working group is investigating OCF and several other IoT topics to determine if there is work for SCTE to do to provide guidance for its members.

Basic Architecture

The prospect of the Internet of Things is one of the most exciting prospects to come along in computing for some time. Science fiction has long anticipated this development, but it has always been too demanding as an application, too expensive in terms of resources, or too impractical in terms of size or speed. All of those barriers are now coming down. Advances in hardware have reduced size and cost. The size reduction has in turn increased compute capability and improved speed to the point that a sophisticated computing device can be about the size of a piece of confetti and process data in essentially real time.

What remains to do is build the infrastructure that can combine this technology with cooperation between organizations so that the same scale and benefits of the Internet can be realized in the Internet of Things. The true benefit of the Internet of Things comes from the combination of computing and communication. The communication is only useful if the various devices are speaking the same language. This is the role of standards.

1. Data Modelling

In defining how the Internet of Things should communicate, it is important to firstly understand what things will be communicating. For the most part, these “things” are real-world devices like lights, refrigerators, lathes and carburetors. Standards are often tripped up by disagreeing on how to define these real-world objects. However, the very fact that the objects exist in the real world implies a certain compatibility between any of the various ways to describe them. This is the fundamental principle behind common data modelling. If modelling method “A” describes an object and modelling method “B” describes the same object, there must be a mapping between “A” and “B.” Similarly, if model “C” describes the same object, there must also be a mapping between “A” and “C,” and by the transitive property between “B” and “C.”

Of course, another problem between standards is what they choose to model. Some standards may only model home appliances while others may concentrate on healthcare devices. It’s hard to map a digital blood pressure monitor to a stereo system.

One way to overcome this problem is to map things at a more atomic level. Both a blood pressure monitor and a stereo system have a switch and a display. If you take all of the atomic components of each system and map them to a common model, you can get some basic interoperability even between items as different as a blood pressure monitor and stereo system.

Establishing a common data model that is a superset of all the atomic resource that are used to compose the devices of interest is a logical way to create general compatibility. Each complex device is just a collection of all the atomic resources it contains. Similar items (like refrigerators) from different manufacturers may have different features, but will still share a majority of common atomic resources like lights, thermostats, switches, etc.

By defining mappings between device representations in any standard and the common data model, a very complete interoperability model can be established.

2. RESTful Architecture

Once a common data model has been set up, the interactions with the data model must be defined. This is where RESTful architecture comes into play. An extended RESTful model can be defined by the following actions: Create, Read, Update, Delete and Notify (CRUDN). In other words, device models can be created and deleted. Their state can be read and set (updated) and devices can send notification if an anticipated event occurs. A very large number of real-world devices can be defined, observed and operated using only these principles.

Table 1 - CRUDN view of a RESTful interface

CREATE	A resource must be created before it can be used. PUT is normally used.
READ	Read allows the current values of the resource to be determined. Get is normally used.
UPDATE	Update allows the current values of the resource to be set. Generally, POST is used so the elements of the structure can be written selectively.
DELETE	Delete is used if the resource is no longer needed.
NOTIFY	Notify is used to get information initiated by the server. It is usually communicated using an OBSERVE function or a Publish/Subscribe method.

3. Security

One of the biggest fears generated by the Internet of Things is security. If hackers can already sabotage your hard drive or steal your passwords, imagine what damage they could do if they had access to the door locks on your house or could control your oven. This is why a security strategy must be built-in to any IoT ecosystem from the start. The security system must not only employ the latest security technologies, but must also anticipate that it will be hacked and have a plan for how to survive the hack and update its own systems. This sort of security can not succeed if it is “added” after the system already works. It must be designed in from the start.

4. Bridging

Standards succeed when they are agreed upon by the key implementers in any ecosystem. This is why cellular phones can call other cellular phones (it wasn’t always that way). One way to make sure everything works together is to get all the key players to choose the same solution. While attractive, this rarely works quickly or without several standards failing the test of time.

Another way to do this is to define bridges between standards. By using the common data model described in section 1, a mapping can be described that maps another standard into the common data model and out of the common data model. If several standards define this two-way mapping, they can all communicate by transitioning through the common data model. A bridge is a device that implements these two-way mappings. By using a bridge, a standard can take advantage of the common data model without the need to adopt it or convert millions of deployed devices to use it.

Organizational Pillars

The architecture described in the sections above provides the theoretical construct for interoperability, but a standard is always open to interpretation. Even a well written standard can be understood differently by different brilliant engineers. In order to ensure interoperability, more is needed. Interoperability can be better enabled by building an ecosystem with three pillars.

5. Open Standard

The first pillar is an open standard. A standard defines as unambiguously as possible all of the details for implementing the architecture. Standards use very precise language and lots of coding examples. An open standard is developed in view of many critics and reviewers. Eventually, it is made public so anyone can access it for review, implementation and improvement. An open standard also generally has very generous usage terms including minimally restricted use, fair and often free licensing terms and a regular process for finding and correcting errors. An open standard is a key element of a maximally interoperable system.

6. Open Source Implementation

Having an open standard is a good start, but relatively few people have the patience or motivation to read a standard and write compliant code for it from scratch. That’s one reason it’s important to have an open source implementation. With open source, the code that implements the standard is created, tested and shared by a community. By starting with a solid code base, individual developers can begin to implement a new product with basic code that already works. They just need to add the features that are required by their new product. With many people reviewing and using the open source code on a regular basis, errors are more easily discovered and corrected. The community works together to maintain and improve the code base to the benefit of all.

7. Certification Test Tool

The final organizational pillar is the certification test tool (CTT). The CTT tests the code to make sure it is compliant with all the requirements laid out in the open specification. By automatically testing each specification requirement, any implementation (private or open source) can be verified to be a valid implementation of the specification. If a device passes the tests in the test tool, it should logically work with other devices that pass the test tool.

By using each of the three organizational pillars, a standard that enables interoperability theoretically, can test a “canonical” open source implementation and verify it is compliant with the specification. The CTT can also test other private implementations. The combination of open source, open standard and a CTT provides multiple checks. When errors are discovered, they can be tracked to errors in the implementation, errors in the CTT, or even errors in the specification.

Additionally, “plug fests” can be held where products that are expected to be interoperable by virtue of passing the CTT can actually be tested with other products that also pass the CTT. This is a nice final test to verify interoperability.

Tools and Support

The features described to this point help to ensure that interoperability and security can be developed across multiple products and ecosystems. However, if it is too difficult to build these interoperable products, nobody will build them. This is why it is important to develop tools and a support system to help product developers.

8. Crowd-sourcing Data Models

One of the best ways to help developers is to provide a complete resource repository of atomic data models. While this can be done in a number of ways, one of the most efficient it to have a common place where anyone interested can create a model and submit it. Of course, this would quickly become unwieldy without a bit of control.

One way to manage this would be to have an online tool where users could log in and make submissions. If these submissions could then be reviewed by a team of experts before being accepted, consistency could be assured and duplication could be avoided.

If this tool also allowed for submission of mappings between different ecosystems and the common data model, there would be a single authoritative source for data model resource interoperability. Furthermore, the accuracy and completeness of these models would be encouraged by support of the community that would use them.

9. Support for Multiple Platforms

One of the real benefits of the Internet of Things is the vast variety of devices that can be built on numerous ecosystem platforms. For some products, a minimal platform is required that can run on a button-cell battery for several years. For other applications, a more capable platform is required that can process video signals or respond instantaneously.

These platforms are not going to all support the same operating system or state machine. They won’t support the same programming languages. This is why it is important to have an architecture and implementation that can support a wide variety of platforms. It is also important to maintain a number of these platforms to give programmers and product developers a head start.

10. Tutorials and Developer Support

Another critical tool for developers is an example. A working example is far more useful than several pages of specification text. Moreover, a tutorial that takes a developer step-by-step through the development process is much more valuable than a simple example.

With the right development tools and a good step-by-step example, a programmer can be productively programming in a couple of hours rather than a couple of weeks.

Open Connectivity Foundation

Now that we've laid out the basic requirements for a secure and scalable IoT system, let's look at the Open Connectivity Foundation (OCF) to see how well it stacks up to the principles for interoperability that have been described. The OCF is an open standards organization with over 300 members worldwide.

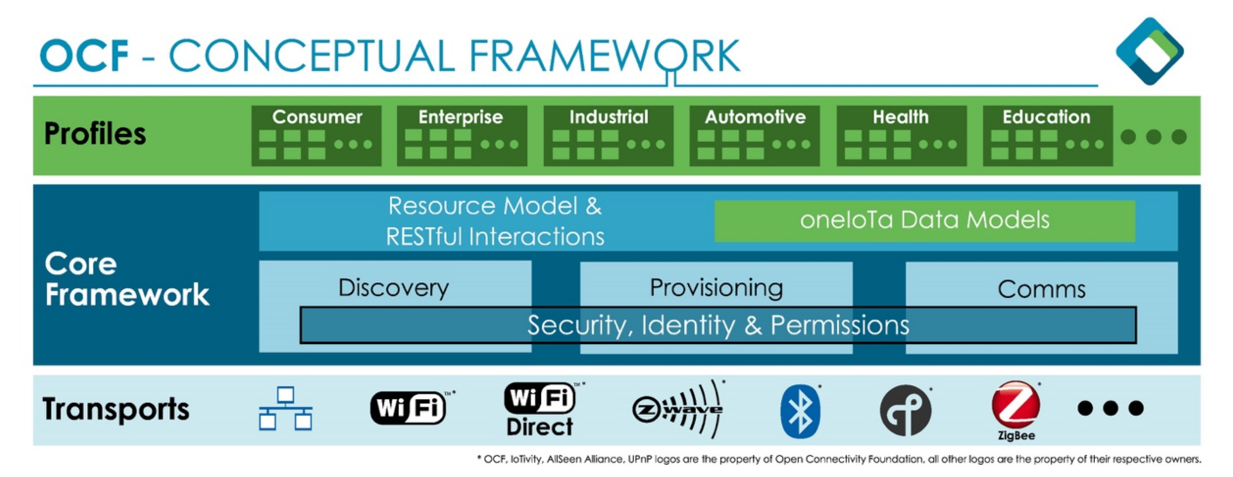
OCF - CONCEPTUAL FRAMEWORK
Profiles
Consumer
Enterprise
Industrial
Automotive
Health
Education
Core Framework
Resource Model & RESTful Interactions
oneIoTa Data Models
Discovery
Provisioning
Comms
Security, Identity & Permissions
Transports
Wi Fi
Wi Fi Direct
Z-Wave
ZigBee
* OCF, IoTivity, AllSeen Alliance, UPnP logos are the property of Open Connectivity Foundation, all other logos are the property of their respective owners.

Figure 1 - OCF conceptual framework

11. Common Data Model & RESTful Architecture

OCF uses a common data model comprised of atomic resources (currently over 100) defined using JSON schema for payloads and Swagger files describing a RESTful interface. OCF uses a CRUDN set of actions on the atomic resources and constructs complete devices as collections of resources. The interface for a device is completely described by the device data model. A "client" or control point interface can operate a device by introspecting the device description and using the CRUDN interface on the device data model.

12. Security from the Start

Robust communication security is designed into OCF from the start. At its most secure, OCF uses a public-key infrastructure with credentials installed into the device at manufacture time. Additionally, OCF uses link-based security. Insecure ecosystems can interface with OCF, but capabilities of these insecure ecosystem will be limited.

13. oneIoTa and Derived Models

oneIoTa (oneIoTa.org) is an online tool for crowd-sourcing resource data models. It is a basic Integrated Development Environment (IDE) that includes back-end processes for submitting and reviewing data models with groups of experts. oneIoTa provides the definitive collection of OCF data models that comprise the common data model. It also contains resources from other organizations like AllJoyn and UPnP. Other organizations are encouraged to contribute their data models.

oneIoTa is also the repository and tool for "derived" data models that describe the mapping between other ecosystems and the OCF common data model.

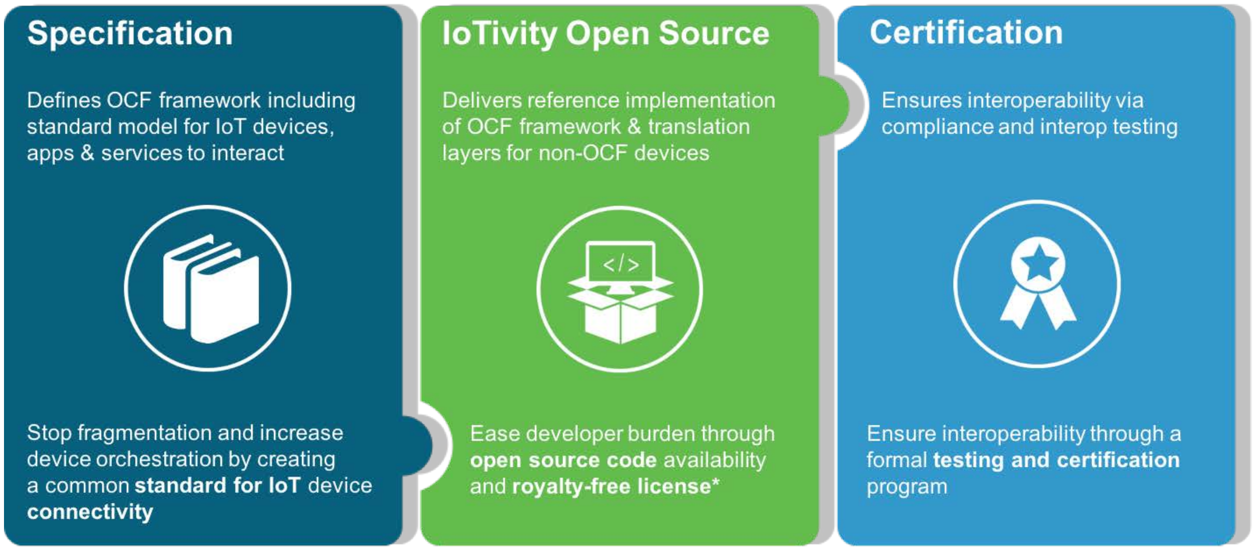
Specification
Defines OCF framework including standard model for IoT devices, apps & services to interact
Stop fragmentation and increase device orchestration by creating a common standard for IoT device connectivity
IoTivity Open Source
Delivers reference implementation of OCF framework & translation layers for non-OCF devices
Ease developer burden through open source code availability and royalty-free license*
Certification
Ensures interoperability via compliance and interop testing
Ensure interoperability through a formal testing and certification program

Figure 2 - Three pillars of OCF

14. IoTivity

IoTivity is the official open source implementation of the OCF standard. At each release event, IoTivity will provide a complete implementation of OCF that passes the certification test tool. IoTivity is managed by the Linux Foundation and funded by OCF. In addition to a complete implementation of OCF, IoTivity includes a number of examples and implementations for several platforms.

15. Certification Test Tool and Authorized Test Facilities

The OCF certification test tool (CTT) is an automated test tool that implements a complete test of OCF at each release event. CTT is under continual development. IoTivity must pass the CTT at each release event. CTT is also the tool that is used by OCF authorized test facilities to certify real products.

16. Developer Community and Tools

OCF has a marketing group that provides resources and instruction events to encourage development and support for devices based on OCF. The Tools Task Group in OCF will develop and distribute tools to assist developers in the creation of OCF products. OCF also encourages independent groups with different interests to support OCF. Implementations for particular ecosystems, platforms and applications adds to the diversity of the development community.

17. Current Status

OCF has not suddenly solved IoT interoperability, but progress is being made. OCF currently has over 350 members and liaison relationships with about 20 other standards organizations. OCF version 1.0 will be publicly released this fall. Each OCF biannual release will include synchronization between the open standard, the open source implementation (IoTivity), and the certification test tool. There is a well-defined process for continual introduction of new vertical markets and associated use cases. Mainstream devices that are OCF native will start showing up this Christmas along with bridges to existing ecosystems. There is still a long way to go, but progress is being made.

Conclusion

In order to get the full benefit of the Internet of Things, a maximal set of things must work together regardless of development platform or underlying ecosystem. The principles described in this paper can help deliver on the promise of IoT by creating a scalable system that can describe virtually any product and interoperate with virtually any other ecosystem. Furthermore, the three pillars of an open standard, an open source implementation and a certification test tool ensure that the implementations of OCF meet the expectations of the theory. Finally, the support community around OCF aims to ensure that OCF-based products can be successfully implemented as easily as possible.

Creating an Internet of Things that parallels the opportunity of the Internet is no small task and the success of OCF is in no way guaranteed. However, by using the interoperability principles described in this paper the chances of success for OCF are improved substantially.

Abbreviations

CTT	Certification Test Tool
OCF	Open Connectivity Foundation
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
ISBE	International Society of Broadband Experts
ISO/IEC	International Standards Orgainzation/International Electrotechnical Commission
SCTE	Society of Cable Telecommunications Engineers
W3C	World Wide Web Consortium