API-fication of the Internet of Things

Turning Things into Services

Part 1

machineshop

White Paper

Published April, 2015

www.machineshop.io

@machineshop.io

machineshop

Purpose of this Whitepaper

We wrote this white paper to engage "makers of things" in a discussion and to challenge them to flip their point of view 180 degrees. It is written to help you realize that the Internet of Things is not about 'things' at all.

For your customers and their developers, the Internet and Things is merely the starting point of their journey. It's where the conversation just begins. This is a conversation or exchange between physical end points and applications that is entirely too complex and unnecessarily so.

Your Thing is Connected to the Internet - So What?

Your customers really don't care about your hardware, per se. Your customers care about the real world, real time data, information and events that connected things can reveal to them and their applications. So the fact that you've helped connect operational assets to the Internet is important, but really just table stakes and the starting point in the big picture.

The fact is that many organizations have thousands, perhaps millions of 'things' connected to their networks and to their applications. Developers, architectures, managers and C-level executives must provide the resources – time and money – to understand all the data coming from all the devices and over many different networks.

Just Because You're Speaking Doesn't Mean I Understand You

If I'm observing real world events as they occur I could see temperature rising, pressure dropping, speed increasing and a system that is on or off entirely. But hardware vendors take human understandable, real world events and convert them into protocols, languages and data structures that humans can't understand.

Just because connected things might effectively gather and transmit terabytes of valuable data in a common protocol, it doesn't mean that users or their applications understand a single bite of it. Sure, vendors provide documentation, they do rely on standard formats and protocols but there is still a learning curve for every vendor, every device, etc. Almost no developers went to school to learn MQQT, COAP or Vendor X's proprietary data format – and there are thousands of these out there.

And painfully, we know this is not a one-time task. Hardware vendors optimize firmware and tweak protocols and data formats all the time. This creates additional upstream work for developers and can be disruptive to users of their applications. In fairness, end users aren't sitting at the terminus of the Internet with a catcher's mitt ready to receive data and events. Applications are at the receiving end and, frankly, they don't speak a human understandable language either.

A World of Services - the REST API

Why can teenagers build apps that integrate data and functionality from ESPN, Facebook and the New York Times in a matter of hours? Simply, because all of these sources of content and application functionality are exposed as simple and standard Web Services, or a standard mechanism to support interoperable machine-to-machine interaction over a network. Web services have evolved over the past several years and the REST API has emerged as the dominant, almost completely universal way that applications, content and services interact over the Internet and World Wide Web.

REST API stands for Representational State Transfer (REST) Application Programming Interface (API). Today, REST APIs provide the simplest way for connected end points (devices and systems) and business objects (applications and databases) to simply interact in a very controllable, standard way that is almost universally understood by anyone building applications – whether enterprise or consumer.

Today, there are literally trillions, with a T, of Internet-based interactions occurring via REST APIs. There are dozens of companies that have built their entire business around providing a set of services via APIs. They have no application; they are simply a set of APIs that provide tremendous value to developers and their applications. [Full disclosure, MachineShop is one of those companies].

Few readers will have heard of Twilio or Stripe but between the two of them they literally process nearly a trillion API transactions every year – and they monetize every single transaction. Though both companies are private, their most recent combined valuations are over $2B with revenue that's a fraction of that. Salesforce.com generates 50% of their revenue, or $1.5B through monetizing APIs and Expedia generates 70% of their revenue through APIs, or $1.8B.

There are two key points here. First, web services, specifically REST APIs, have become the way the technology world is communicating. Second, these 'services' are highly valuable and valued in the marketplace.

Transforming Your Awesome Proprietary Protocol and SDK

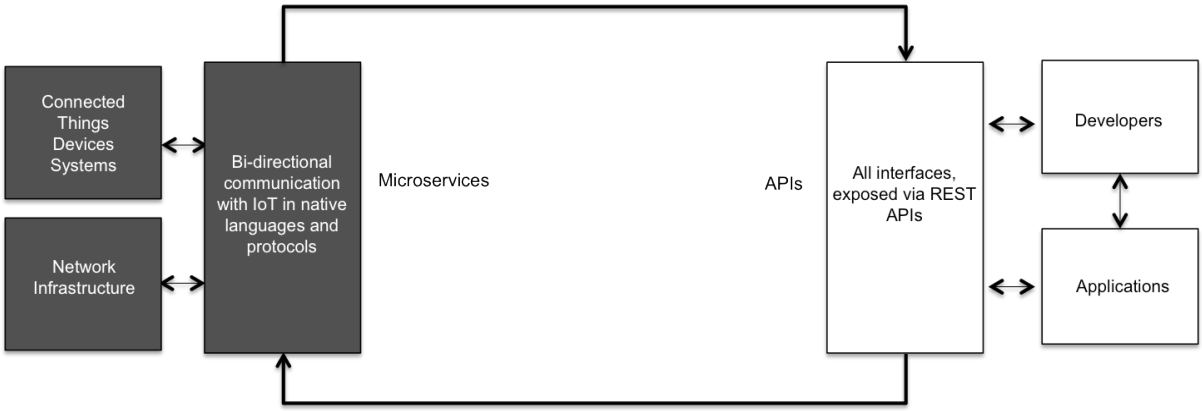
If you have come this far in the White Paper, you might be intrigued or even

committed to the idea that web services and REST APIs should become part of your customers' experience with you and your hardware product. The work of an application developer goes beyond understanding the device data report format but also the protocols necessary to receive or transmit those messages.

vendors to either natively adopt web services and REST APIs as their standard method of interacting with applications or they can simply provide a service that transforms the native protocol into REST APIs without changing the underlying protocol. These are not mutually exclusive options and can be a simple

Let's look at an example of a typical connected "thing":

A connected device is configured to publish its information over an MQTT channel requiring an application-side MQTT client to handle the subscription and receive the data from the device.



There are newer transport protocols such as MQTT, COAP, LWM2M, HTTP and standard TCP/UDP.

Each of these protocols has specific methods that must be employed for use within an application. Of course, like most standards, there are also variations that must be managed between vendors.

Even more challenging than the protocol for the data transmission is the wide variety of message formats sent over these transmission methods.

Protocol Transformation or Agent?

There are some IoT platform vendors that solve the protocol normalization problem by requiring specialized agent software on the device. On most legacy devices, this is just not possible. Even where possible, hardware vendors and their customers have committed the egregious sin of lock-in when they adopt this approach.

The better alternative is for hardware

matter of evolving the strategy over time.

The API-fication of IoT

Transforming outdated, static, proprietary SDKs or protocols does not put hardware vendors in competition with their customers or integrator partners.

On the contrary, nothing will speed time to market, reduce complexity and cost more than standardizing the way developers build on top of your hardware than adopting an API-centric strategy.

Providing device information and interfaces through easy to understand REST API services allows developers to focus on the actual data and control of the device and not on the protocol or transmission method. Effectively, the device and its control become as simple as interfacing to a standard REST API which can be integrated into almost any application.

MQTT generally publishes information over what is called a topic.

```
Device Name: 0013430C9807
Topic: BB/0013430C9807/data
```

The data that is presented over this MQTT topic is generally sent in some sort of proprietary message format. In this case, the message being published is "s\":9,\"t\":\"2015-04-10T22:16:30Z\",\"q\":192,\"c\":3,\"do1\":false,\"do2\":false

While it is certainly possible for the application to implement the required services and interpret this message format, imagine the load on the application if there were 1000's of these devices, each with slightly different versions of the message format.

An API-centric approach allows specific, authorized services to alleviate this problem by handling message reception and translation of the proprietary format into something that can be

consumed and presented in a standard manner to the application. Using standardized JSON (JavaScript Object Notation) message formats, the application can either poll for messages, subscribe to a stream of translated messages, receive them directly from our platform using push technology or make a REST API call.

In the case of this example message, it was converted to a report that looks like this:


```
{
        "id": "55284bed50e7b0394f002f93",
        "created_at": "2015-04-10T22:16:47Z",
        "data_source_datetime": "2015-04-10T22:16:30+00:00",
        "data_source_id": "54f61cf150e7b0e722000cde",
        "mqtt_topic": "BB/0013430C9807/data",
        "payload": {
        "sequence_number": 9,
        "timestamp": "2015-04-10T22:16:30Z",
        "quality_code": 192,
        "configuration_index": 3,
        "digital_out_1": false,
        "digital_out_2": false
        },
```


Instead of trying to read directly from the device and dealing with the complexity of that process, the developer's application makes a simple API call to get the information needed for the business process. In addition, new information can be appended to the information to indicate method of reception or possible time received.

Now the application can understand the message in more standard terms like timestamp, quality code, configuration index, etc. No need for any conversion of the data. No need to manage multiple protocols and versions, multiple transmission methods.

In a similar manner, the application can publish back a message for this device. Using a REST API POST with the same field descriptors as in the message above. For example, if the device accepts a command to turn digital_out to TRUE, the developer could use a JSON body with digital_1_out:TRUE and the outbound services would convert the message to the appropriate format and send it back using an MQTT server to the device client MQTT topic.

APIs Are Not All Created Equal

A word of caution and pre-emptive push back. APIs are nothing new; they have been around for literally decades. But only in the most generic sense. After all, API simply means an interface exposed to an application. Nothing new about that.

One of the first universally adopted API formats was SOAP for Simple Object Access Protocol. SOAP APIs served a great purpose for years but are not particularly well suited for the World Wide Web and certainly not for asynchronously connected things. So before sticking out your chest and claiming moral high ground for presenting SOAP APIs to your customers, you still have work to do.

Very important - there are also a number of considerations around creating and managing APIs that are not addressed in this White Paper. They are the primary subject of "API-fication of the Internet of Things, Part 2". As services, APIs must be carefully authenticated and metered so the provider and consumer of these services have a very clear understanding of the contractual relationship governed by the API.

A Thing as a Service

There would be no Internet of Things without things. As a hardware vendor, there is just cause for excitement about the growth potential of this market for all. The purpose of this white paper was to help hardware vendors understand the importance of simplifying application development and management by exposing interfaces to hardware, and their data, through standard, universal services, or REST APIs. Not only will 'things as services' make life easier for your customers, it will provide valuable information to product management and executives about the ways your hardware product is utilized, which features matter and more. In some cases, this may also provide a new source of revenue.

By creating and managing interfaces through web services, vendors can monetize a valuable link between hardware and application software. APIs are generating billions of dollars of revenue for hundreds of Internet-based companies today. As the foundational layer of the IoT, there is no reason hardware vendors can't join the party.

MachineShop provides the on-ramp to the Internet of Services where trillions of transactions will occur between billions of physical things and their interactions with other systems, applications and people. MachineShop's public or private Services Exchange allow customers to subscribe to thousands of unique, managed API-centric services regardless of whether they are developed by MachineShop, its customers or other third parties.