English

Smart API

The benefits of using linked data with IoT

ASEMA

Table of Contents

1. Improving the ways to transmit and store data

1.1. The need for improved data formats in IoT projects

Data matures like wine. Applications and hardware mature like fish.

If you work as the CIO of any midsized or larger organization or in a small firm trying to partner with others, you'll know how hard it is to push people into a joint direction under the pressure of conflicting business needs and technological demands. And yet, unless some common way of working is established, nothing really works - or at least becomes prohibitively expensive to make it work.

But in a domain like IoT which combines so many fields of engineering, from hardware and software to telecommunication and data analysis, it is often hard to find a common ground. Certain hardware may be prohibitively expensive for one business unit while it does not provide enough features for another. Software simple enough for one organization is too insecure for the other. This is the daily reality in corporate connectivity projects.

It is tempting to try to set corporate standards for some of this. For instance, you could require a certain ARM processor in all hardware or a standard wireless networking solution. Or just buy everything from the same place, and dump the problem to the supplier. However, there is a caveat in such an approach. It may be possible, though difficult, to have all business units accept one solution now, but what about the future? Many IoT devices are required to stay operational on the field for decades. As the hardware gets old, there is increased pressure among projects to break ranks later in order to keep up with competition while others are forced to stick with what they have. And whenever you standardize something you don't own, you risk supplier lockup.

The correct solution is to build the common ground on what you do own, the part of the chain that actually is generated by your organization and which describes the operations of your organization alone. This is the data. The key question then is how to actually do that in a way that is truly future-proof.

Smart API builds on a semantic, linked data format. While no data format can be guaranteed to be "it", the ultimate all-capable data set that encompasses answers to all demands days to come, it is currently regarded as the most advanced, the most comprehensive way to describe data. While it may carry a 100% guarantee of being the final answer to any question - nothing does - it is by far the best bet.

Let's explore why.

1.2. What is linked data and what makes it so special?

Linked data is data that describes relationships. What makes that differ from other forms of data? Well, the most common data, the non-linked one, treats data as simple assignment of values. A = 2, color = blue, speed = 20. Such value assignments are the basis of a vast majority of databases that store our digital world today. A database has rows with items and columns that give items values. A product database could for instance have values in columns such as name = bicycle, color = gray, size = medium.

Linked data describes the links between those items. Instead of just an assignment i.e. X `[equals]` Y, the relationship can be anything. Product A `[is sold by]` Pete. Product B `[was bought by]` Mary. Product D `[belongs to the same bundle as]` Product C. While such relationships can of course be modeled into any data, in traditional data relationships are exceptions. In linked data they are the norm. While the distinction may sound subtle, in real implementations they make a big difference. That difference comes from the world of graphs. Graphs are the basic technology for storing linked data.

Graphs are nowadays extensively used in social media systems because social media is all about connections between people and things. Graph technology makes handling such data much easier and faster. Consider a marketing campaign that wants to promote merchandise of actor Dwayne "The Rock" Johnson to his fans. We may know that a user called Pete just bought something from our webstore so a good set of potential buyers could be Pete's friends who also like The Rock. So we want to find

those people from our social media database, like so: "Find all Pete's friends who have recently liked movies starring Dwayne Johnson".

Performing this search would be complex with traditional relational databases that work on simple value assignments. It would involve many rounds of subqueries to that would first find the movies where The Rock is in, then filter out those where his role is the star and then somehow cross-correlate this result with the friends of Pete who have given certain types of likes.

However, with graphs that natively embed such connections, the above would be just one search that can be optimized and indexed to immediately yield a result. Because graphs are designed to handle connections, they are technically superior in describing things in applications that involve such connections.

And Internet of Things is all about connections. Consider an example such as "Find the thermostat that controls the temperature of the living room in the house where Pete lives in". Or something more complex: "Find all the levers that control the power output to analysis equipment at the lab where Pete works in and set them to normal level in case Pete has tampered with them after seeing a movie by The Rock and thinking it is always such a damn a good idea to go to the max". And so forth.

The relationships stored in graphs take us to the three fundamental properties of linked data that make it so powerful in Internet of Things.

1. Contexts

2. Relationships

3. Abstractions

Let's explore these in more detail in the next chapter.

2. Linked data benefits in real life

To highlight how the three properties of linked data - contexts, relationships and abstractions - work in real life, let's have an example use case. Assume we have a company, Excavation Inc, that is hired to dig land at a mining operation. As an operator, it does not own the equipment it uses but instead leases it from three subcontractors, Machines Inc, Equipment Inc, and Tools Inc. The machines are intelligent, partly autonomously operating, and remote controlled using IoT technology.

2.1. Contexts

First, contexts make it possible for us to describe in which context are we talking about data. This makes it possible for automation systems to treat it in different ways in different situations and to unify data from multiple sources. Take the example of maintaining those machines. For preventive maintenance, Excavation Inc has a tracking system that has a bookkeeping of the operating hours of machinery. Maintenance personnel mark the repair work of the machinery there so the system becomes a way to predict the need to replace wear-and-tear parts after a certain amount of hours of operation. To get the operating hours, Excavation Inc downloads the data from the systems of the subcontractors. However, the way subcontractors track the hours is different:

- Machinery Inc: Average daily operating hours = recorded hours per month divided by 30
- Equipment Inc: Average daily operating hours = recorded hours per week divided by the number of business days of that week
- Tools Inc: Average daily operating hours = recorded hours per day, each weighed by the type of load and strain on parts, averaged over a period of five days of operation

Making the same conclusions based on data from Machinery Inc would result in vastly different results than that of Equipment Inc and Tools Inc. For instance, one includes holidays and weekends in the calculation while the others do not. Consequently, the data needs to be unified.

To be able to differentiate the data, Excavation Inc stores and communicates it with a prefix. Instead of having just a variable "AvgOperatingHours", it has three variables: "machinery:AvgOperatingHours", "equipment:AvgOperatingHours", and "tools:AvgOperatingHours". The part before the double-colon, the prefix, tells not only the source of the data, but also its definition in the context specified by the prefix. Technically the variable definition is a link. If a computer would follow the link, it would find a specification that tells the difference between the three measures of operating hours. It can then automatically convert the data to arrive to one uniform set of values Excavation Inc can rely on when it decides to do maintenance on the equipment.

2.2. Relationships

Second, relationships help us point out connected things with ease and precision. Let's assume there is a need to adjust a cooling fan in some of the equipment in the field. Cooling fans are operated by three onboard computers that may be assigned to run any of the connected sensors and actuators inside that machine. To access a computer remotely, the operator needs to go through a cloud system and in this case there are three of those, one for each subcontractor. Remote access requires authentication and a permission from one of the two authentication service providers Excavation Inc uses. Looks like a lot of links!

As was highlighted in the introductory chapter, handling systems where such links exist are the bread and butter of linked data systems. In this case for instance, to get the permission to make the adjustment, Excavation Inc could for instance be required to perform the following search: "find the system where we can get access authorization to the cloud which is connected to the onboard computer which is assigned to manage the cooling fan that controls the temperature of lifting piston W3 on machine BCH-1102". Sounds complex? For graphs it is not.

2.3. Abstractions

Finally, abstractions. The issue with automated equipment is that the more intelligent they become, the more intelligently we need to be able to talk to them. Instead of assuming that they can be given just dumb on/off commands, modern equipment may require instructions in much more abstract format.

Take the example of our piston cooling fan. Let's assume that a mechanic has reported that based on the noise they make during operation, the current fan speed puts too much strain on the bearings of the fans, which may cause malfunctioning in some of the units. So the mechanic team decides to lower the fan speed by 200 rpm. However, the fans are controlled by onboard automation. They don't have just one setting. Instead, the speed is dynamically set depending on various factors. Higher lifting load creates more heat which requires higher fan rotation speeds to cool down. On dry weather the fan can be run at a low speed but under humid conditions the fan not only takes heat away from pistons but also prevents moisture from entering the system, again requiring higher fan rotation speeds. And so forth. So the fan cannot be controlled by just saying "set speed to X rpm". A more abstract command is needed instead.

Abstractions are a part of linked data design, its semantics. Not only can things be connected but also the definitions of things have links. These links make computing systems understand that when we talk about some thing, we actually also talk about some other, more abstract concept. So if we talk about the material of the seats of a car, we are actually talking about how luxurious or comfortable the car is. With links, some data definition can be the subclass of some other data definition which may belong to the same group as other definitions. Car seat material is a subclass of car comfort, for instance. This way linked data can describe abstract concepts. In the case of our example, the concept could be for instance "performance". Instead of saying that fan "speed" should be set to, say 2000 rpm, we could tell the system to set fan "performance" to "moderate".

"Moderate performance" is in the example case an abstraction that could be handled by the onboard computers of excavation equipment. The benefit of such abstractions is that now the computer can operate the fan at variable levels that conform to the specification of the concept. But that is not all. It is now also much easier to program into the onboard computers logic and rules that compensate for the possible changes in other parts of the machinery. For instance, a rule may say that if fans are set to moderate performance, then load warning indicators should warn the control logic of excessive lifting load at 10 tonnes lower levels to prevent pistons from overheating.

Now, many of the benefits of linked data can be replicated with other designs and true, a lot of the features and functionality described above has been implemented in various types of equipment for a long time. Why would it not be, the need is real and tangible. The difference is that linked data offers one consistent, standard and holistic approach to the topic. The tools and designs already take these issues into account so that new implementations are faster to build, more reliable to operate, and easier to connect to each other. With the advances in technology such as connectivity and autonomous operation moving at current speed, old designs quickly become obsolete. This is why solutions built on modern designs such as linked data are urgently needed.