IoT Data Interoperability POC: a Pragmatic Feasibility Proof

1 Introduction and problem statement

The importance of IoT interoperability is widely acknowledged, touted by standards and vendors, and often discussed by industry analysts. The McKinsey report "Unlocking the Potential of the Internet of Things" estimates that achieving interoperability in IoT would unlock an additional 40% of market value. So, how can semantic IoT data interoperability be accomplished in practice?

It does not seem likely to happen through standards - there are too many competing ones in IoT data space, fragmented and conflicting in some cases, with no discernible dominant player. They are mostly focused on intra-specification interoperability among compliant devices, but not across specifications and domains where the estimated 40% of the additional value is.

An alternative top-down approach would be to try to structure a meta ontology from the existing standards definitions. This tends to be rather cumbersome in practice and it ran into problems with semantic web efforts. It is too slow to cope with evolving IoT standards and the explosive growth of types of IoT devices.

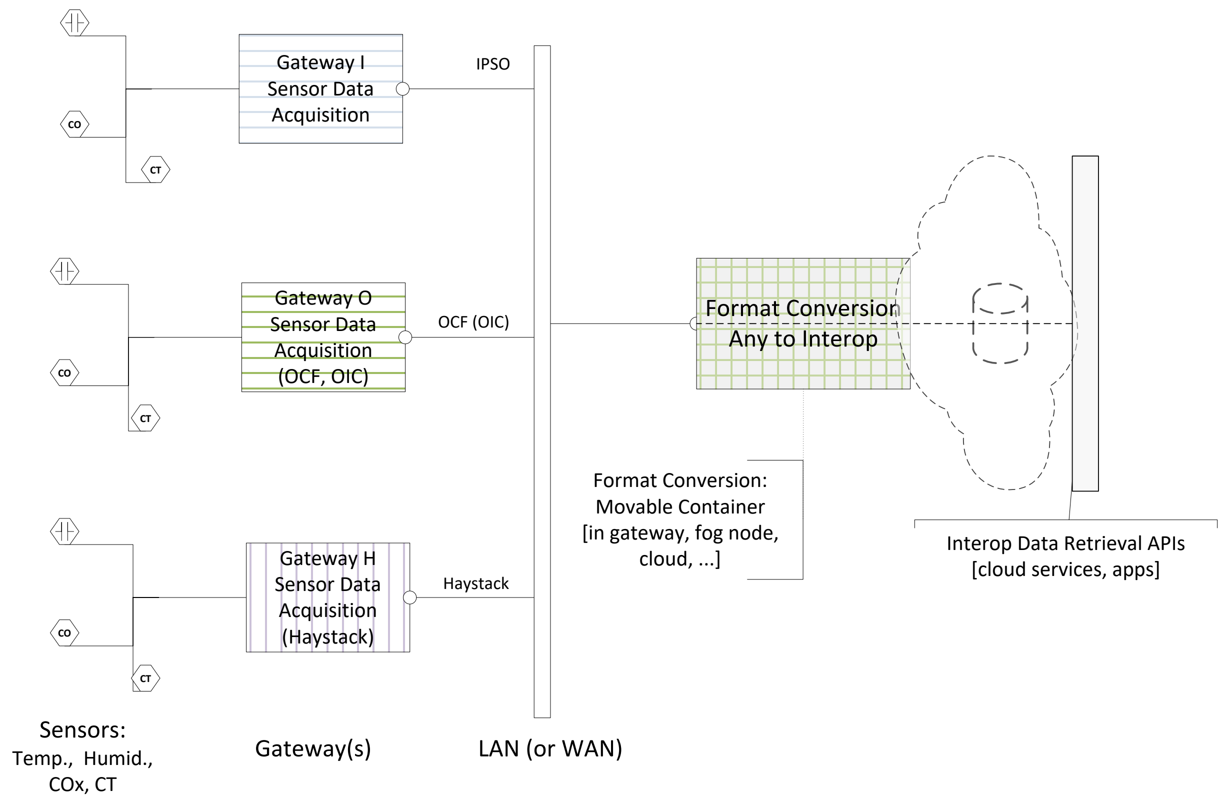
We explored an alternative pragmatic approach to cross-standard interoperability by selecting some representative IoT standards and prototyping a translation into a common interoperable data format. This proved to be feasible and practical, as well as instructional, and we are in the process of expanding and promoting it as an added activity to leading IoT standards bodies.

2 Implementation: data interop POC

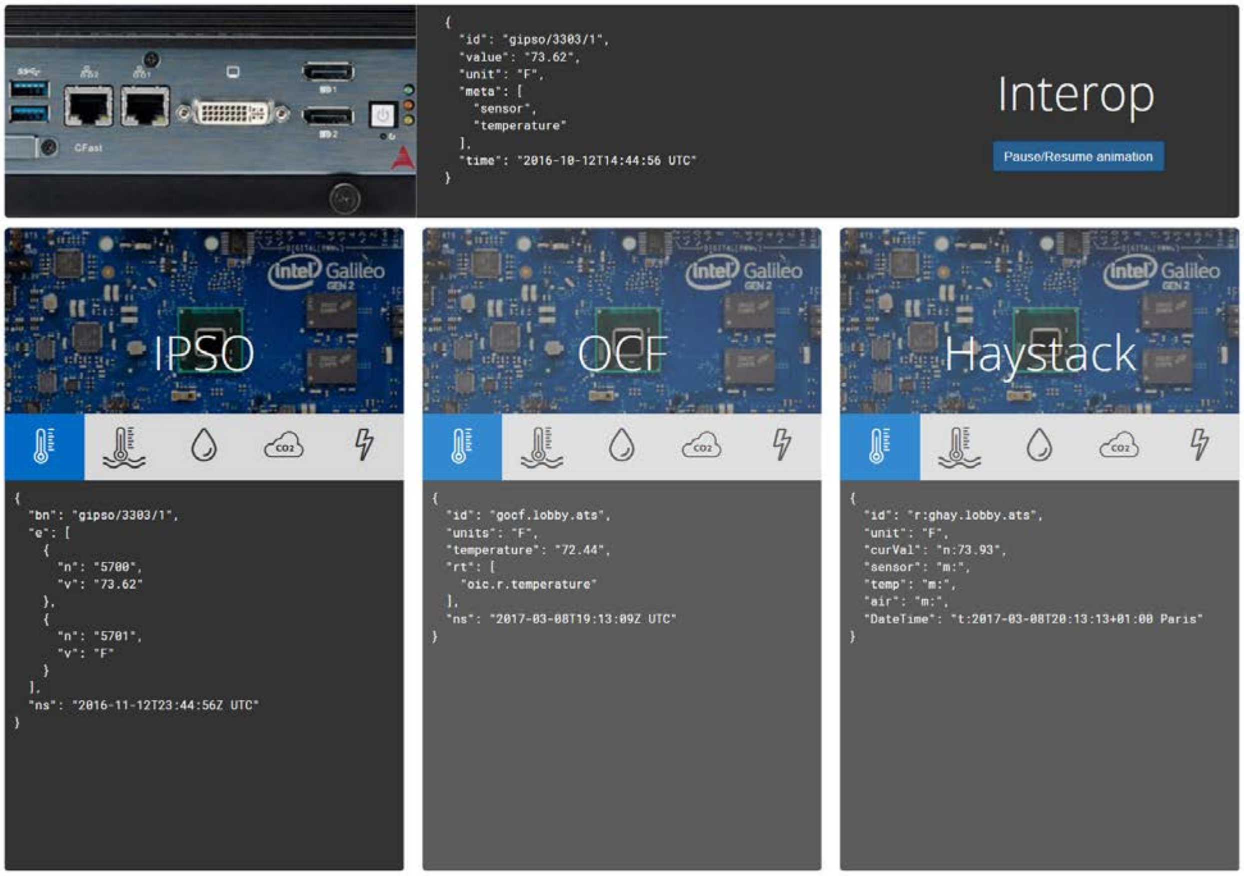
To explore the feasibility and complexity of IoT data interoperability in practice, we created an interoperability proof of concept (POC). The basic idea was to mimic real-world IoT diversity by having a collection of sensors whose data are encoded in different standards and converted in real time to a common format to be consumed by IoT applications and services in the cloud.

The setup consists of 3 gateways acting as sensor data aggregators (Intel Galileo boards), each with a set of 5 sensors that measure same physical entities – air temperature, (simulated) chilled-water temperature, humidity, CO2, and electrical current flow. Sensor selection was driven by an actual smart-building engagement and use case. Specific sensor types were chosen for their wide use in different domains, such as industrial, weather, commercial buildings, and homes.

The code running on each gateway samples sensor data periodically, encodes it in one of three popular IoT standards - IPSO, OCF and Haystack respectively - and sends data to the fourth gateway that is acting as the format-conversion unit. The format-translation gateway receives data encoded in specific standards and converts/translates them to a common interoperable data format which is the same regardless of differences in the input data, thus providing consistent interface to cloud services, such as analytics or ML. Note that the format-conversion function could be placed anywhere in the data path before the data-retrieval APIs, e.g. as a container service running in a gateway, fog node or cloud. In our implementation, we placed this function in a separate larger gateway to make the demo self-contained. We also used that gateway to host and drive the POC UI (live link) function.



1 Interop POC System Architecture



```
{
  "id": "gipso/3303/1",
  "value": "73.62",
  "unit": "F",
  "meta": [
    "sensor",
    "temperature"
  ],
  "time": "2016-10-12T14:44:56 UTC"
}
```

```
{
  "bn": "gipso/3303/1",
  "e": [
    {
      "n": "5700",
      "v": "73.62"
    },
    {
      "n": "5701",
      "v": "F"
    }
  ],
  "ns": "2016-11-12T23:44:56Z UTC"
}
```

```
{
  "id": "gocf.lobby.ats",
  "units": "F",
  "temperature": "72.44",
  "rt": [
    "oic.r.temperature"
  ],
  "ns": "2017-03-08T19:13:09Z UTC"
}
```

```
{
  "id": "r:ghay.lobby.ats",
  "unit": "F",
  "curVal": "n:73.93",
  "sensor": "m:",
  "temp": "m:",
  "air": "m:",
  "DateTime": "t:2017-03-08T20:13:13+01:00 Paris"
}
```

2 Interop POC UI Screen Shot

The reading highlighted in the figure shows the IPSO encoding for a temperature sensor whose ID is gipso/3303/1. Other encodings of the equivalent sensor, OCF, Haystack and BMP are greyed out. While there are obvious differences in encodings between standards, the interop format is the same for all three, with only sensor IDs being unique as they refer to different physical sensors attached to different gateways.

3 Conclusions and next steps

The POC demonstrated that our ground-up approach to translating individual standard definitions is feasible and practical. In retrospect, this stands to reason because all standards are modeling the same physical entities, real world "objects", thus resulting in (mostly) comparable abstract object models. Coding of the translation function was further simplified by the fact that all three standards provide variants of the popular JSON [key, value] form of data serialization.

On the negative side, we experienced the drawbacks of non-isomorphic and rigid modeling of smart objects in several standards. In some cases, we were forced to provide superfluous data fields to satisfy arbitrarily clustered resource properties, in others we were unable to express the available numerical sensor readings and had to use Boolean values instead. We'll communicate these findings and other insights gained from the POC to the related standards bodies.

We also implemented an expanded version of the POC with the additional proprietary sensor data-format to test the effectiveness of the proposed approach beyond standards. The outcome was successful but it is not reported for confidentiality reasons.

Being positively and strongly encouraged by this experience, we intend to work with several standards bodies to promote adding of semantic data interoperability to their charter and deliverables.

Acknowledgments

This work was partially funded by Intel corporation. POC concept, architecture and system design by IoTsense, implementation HTEC group.