Thing Description as Enabler of Semantic Interoperability on the Web of Things

Sebastian Käbisch, Darko Anicic
Siemens AG - Corporate Technology

Internet of Things (IoT) will connect devices, facilities, and networks with advanced sensors, controls and software applications. Combined with the power of analytics, artificial reasoning, automation and deep domain expertise, it has the potential to increase performance gains across global industry sectors. Yet the vision of Internet of Things is challenged today by a number of issues, few of which are associated with data silos, machine interoperability, automated resource discovery, unambiguous interpretation of IoT data, smooth engineering and maintenance of IoT systems and many others. Hence a few challenges that motivate the need for IoT Semantics are:

Semantic interoperability - ensures that IoT data can be comprehended unambiguously by both human users and software programs across different platforms and domains. It offers interaction between heterogeneous Things on a higher level of abstraction.

Interpretation of data and knowledge – makes the data generated by Things understandable by machines and humans, without prior knowledge about Things that produced them.

Unambiguity – deals with an unambiguous meaning of Thing's data and properties. For example, it is not sufficient to know that there exists a Thing, but it is important to know what exactly a capability of the Thing is, and to unambiguously understand the data it produces or consumes.

Data/knowledge integration – integrates data/knowledge from multiple vertical domains into cross-domain applications.

Thing Description

The approach of W3C WoT IG, which addresses challenges presented in the previous section, is grounded on *Thing Description* - a semantic description of actual Thing's capabilities and the information on how to access them. Thing Description (TD) is designed to be minimal in a sense that it is applicable to any Thing, regardless of the Thing's size or an application domain the Thing is used in. It is important to make TD applicable for very resource constrained devices such as for example MSP430, ARM Cortex-M3, ESP8266. Also it is important to make TD extensible so that TD is applied to different vertical domains and cross-domain applications. The task force of the TD group has currently identified following main aspects to specify a TD instance:

*Thing's metadata****:***

- Generic information of a Thing (e.g., name of Thing, supported protocols, encodings etc.);
- Extended-able with domain or vendor specific information.

Thing's Interaction Resources:

- *Property:* used to describe Thing properties. A property can be either static or dynamic (e.g., temperature value, fill level water measurement etc.).
- *Action:* invokes actions on a Thing which starts a process (e.g., LED fade in, move a robot, brew a cup of coffee etc.).

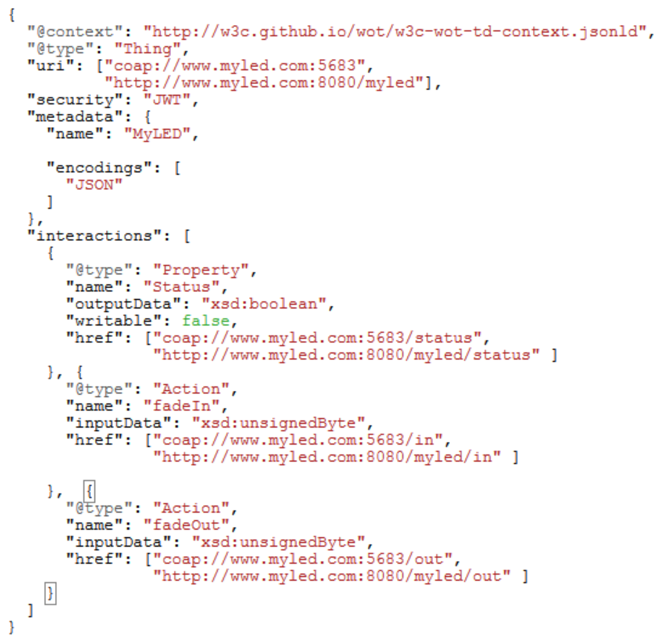
- *Event:* enables an intention to be notified by the Thing on a certain condition (e.g., door opened).

Thing's Communication:
- Protocol and addresses location (e.g., HTTP, CoAP, BLE, etc.);
- Bindings to the interaction resources.

Thing's Security:
- Describing prerequisites to access Thing's resources;
- Protecting TD.

The TD relays on the Resource Description Framework (RDF) as an underlying data model this also enable to be extendable, e.g., to involve domain or vendor specific information. As a current serialization format JSON-LD has been proposed in order to benefit from both the widely used JSON-based format and JSON-LD's concept of *@context* (that provides the mapping from JSON to an RDF model). Since JSON-LD is a text-based representation, a TD may be a burden for resource constrained devices. Based on this, additional binary RDF encoding formats that handle string redundancies well will be also considered in the future.

Figure 1 shows a sample TD of a LED Thing ("MyLED") serving a *status* (on/off) information as a *Property* and two *Actions* *fadeIn* and *fadeOut* that both takes a time input (=unsignedByte). In addition, the Thing supports two application protocols, namely HTTP and CoAP, as well as the used JSON as data exchange format. For addressing the concrete property and action the known *href* is used in the TD. Another important aspect which has to be semantically understood and processed is the data type of Thing's resources. Current working assumption relies on XML Schema data types that are also supported by RDF. This takes the advantage to describe the data in a very precise manner (e.g., use data types relevant for constrained devices (e.g., bytes) and define ranges/limitations etc.).



```
{
  "@context": "http://w3c.github.io/wot/w3c-wot-td-context.jsonld",
  "@type": "Thing",
  "uri": ["coap://www.myled.com:5683",
          "http://www.myled.com:8080/myled"],
  "security": "JWT",
  "metadata": {
    "name": "MyLED",

    "encodings": [
      "JSON"
    ]
  },
  "interactions": [
    {
      "@type": "Property",
      "name": "Status",
      "outputData": "xsd:boolean",
      "writable": false,
      "href": ["coap://www.myled.com:5683/status",
               "http://www.myled.com:8080/myled/status" ]
    }, {
      "@type": "Action",
      "name": "fadeIn",
      "inputData": "xsd:unsignedByte",
      "href": ["coap://www.myled.com:5683/in",
               "http://www.myled.com:8080/myled/in" ]

    },  {
      "@type": "Action",
      "name": "fadeOut",
      "inputData": "xsd:unsignedByte",
      "href": ["coap://www.myled.com:5683/out",
               "http://www.myled.com:8080/myled/out" ]
    }
  ]
}
```

Figure 1 TD sample based on a LED Thing

Extending Thing Description with Contextual Semantic Models

Thing Description offers the basic information about a Thing. For example, "MyLED" is a Thing that has a Property "status" and data that can be either "true" or "false" (see the TD sample above). But if we want to discover a Thing or use it for an interaction with another Thing we need further information. For instance, we need to know what kind of a Thing it is and what additional properties and capabilities it has.

“MyLED” is just a string that a human may understand. For machines it is meaningless. For example, a machine cannot discover “MyLED” when it was given a task to find a sensor of type light, or a lighting device with a certain RGB specification, location and so forth. This information does not exist in its TD. Hence it is not possible to discover the Thing based on it. In many applications we actually discover Things exactly based on such information, and need their TD only after Things are discovered.

Further on, to connect Things in a semantically meaningful way we need again more semantic information than provided in TD. For instance, a temperature sensor from Thing 1 may control air-condition from Thing 2. We are allowed to connect these two devices based on their TDs, also when the connection is semantically incorrect (e.g., the sensor provides temperature data in Celsius and the air-conditioner expects data in Fahrenheit).

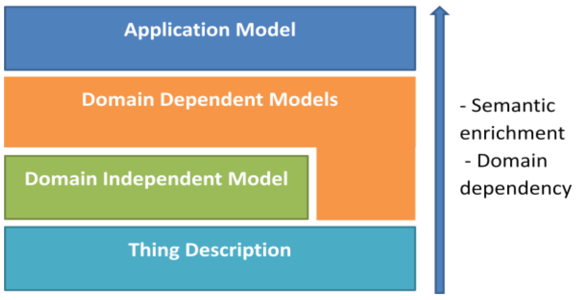


Figure 2 Enrichment of Thing Description with Contextual Semantic Models

Discovery and Thing to Thing interaction are only two examples where we see that TD needs further enrichment with semantic models. Figure 2 shows how Thing Description can be extended with additional semantic layers. For example, TD may be accompanied with Domain Independent Models. Horizontal semantic models are typical examples of such models, e.g., the QUDT Ontologies[1] to provide quantities, units, dimensions and data types, or Semantic Sensor Network Ontology[2] to describe sensors irrespective of a domain in which a sensor is used. Domain Dependent Models is another layer that provides domain specific semantics, also when a model covers few different domains. Examples of such models are eCl@ss[3], IEC Common Data Dictionary[4] and many others available today. Our position is that we need to provide a mechanism which enables TD to utilize existing semantic models relevant for WoT. The question arises as to how existing data models can be mapped against TD and each other to offer interworking. We believe that RDF as an underlying data model offers a solid base for such an integration. This is the reason why we grounded TD on RDF. But the question how to “convert” existing non-RDF data models into RDF ones needs to be discussed. Smart Appliances REFerence (SAREF) ontology[5] is a good example that this is possible. It is an RDF/OWL ontology that facilitates the matching of existing assets (standards, protocols, data models etc.) in the smart appliances domain. Existing standards considered in SAREF ontology are for example: ECHONET, KNX, EnOcean, Z-Wave, OMA Lightweight M2M, UPnP and others. Originally data models of these standards were not provided in RDF. However, thanks to SAREF project, they are now available in RDF/OWL and can be used in semantic WoT applications.

[1] http://www.qudt.org/
[2] https://www.w3.org/2005/Incubator/ssn/ssnx/ssn
[3] http://www.eclass.eu/
[4] http://std.iec.ch/iec61360
[5] https://sites.google.com/site/smartappliancesproject/ontologies/reference-ontology