The Smart Object API in the IoT Infrastructure

Data Models for the Internet of Things #5,

Michael J Koster
September 9th, 2012

*The goal is to create a web ecosystem of **sensing**, **reasoning**, and **action** around the Internet of Things.*

The Smart Object API is a Semantic Web application for the Internet of Things. A RESTful web object encapsulation of semantic elements and real-time data properties, the Smart Object API provides pluggable live linked-data interaction between application software agents and IoT endpoints, sensors and user devices.

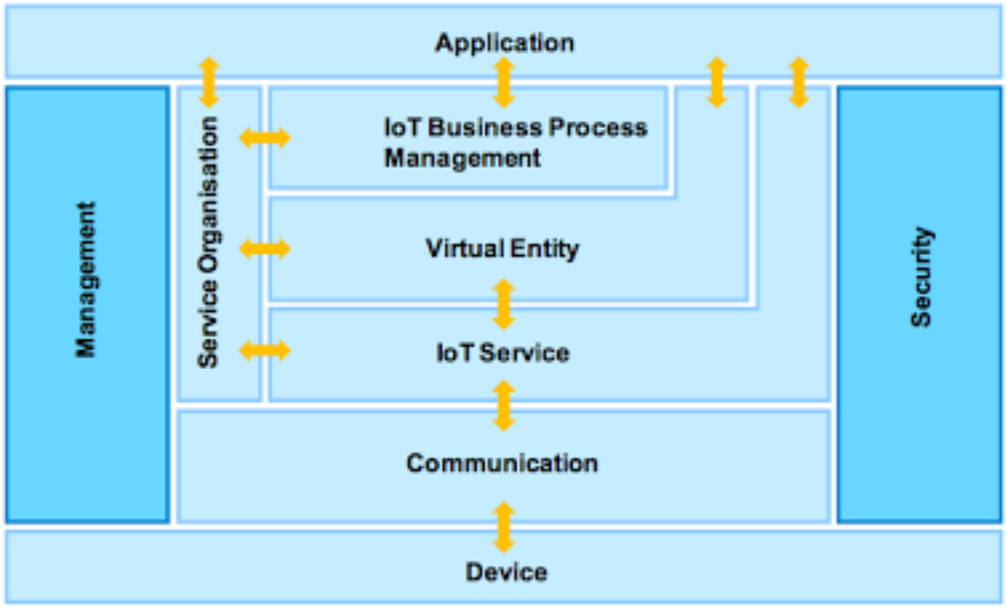
In the previous posts, I introduced the concept of a common data model for the Internet of Things based on emerging WWW technology and design patterns. I propose a Smart Object API as the basis of one such data model.

In this post I want to discuss how the Smart Object API would fit into the larger IoT infrastructure. There are a number of perspectives to consider, among them supporting service infrastructure, deployment patterns, integration with other stacks and standards, and specific build-out of the Smart Object pattern itself.

For a reference model of the Internet of Things, I refer to the EU IoT Architecture Project's (newly revised) Architecture Reference Model, which is a broadly inclusive description of IoT architectures.

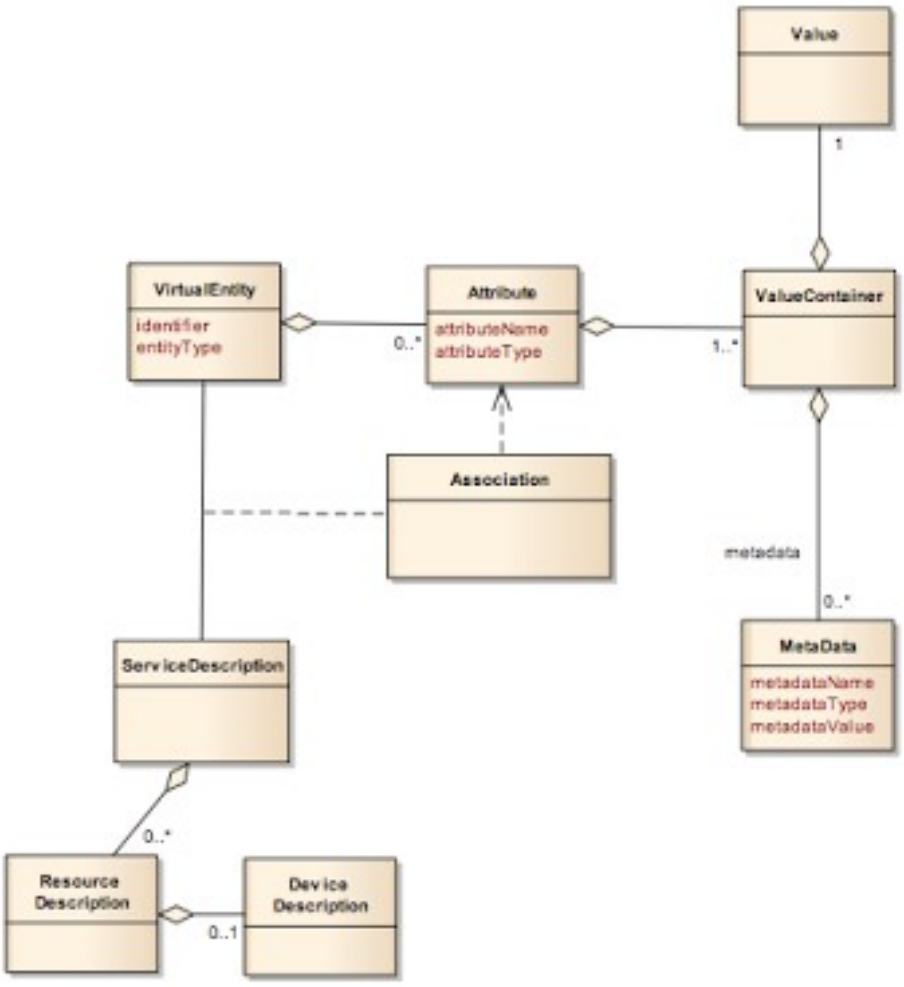
High-Level Infrastructure

For a view of the overall IoT infrastructure, and how the Smart Object API fits in, a good summary is in the IoT-Architecture document, D1.3 Architecture Reference Model, Figure 15, Functional View:



The Smart Object API defines a Virtual Entity, Application Interface, and IoT Service interface. The Smart Object framework will additionally define some common application agent patterns and a general service framework.

The IoT Information model, figure 13 of the same document, further defines the Virtual Entity and it's component relationships. The Smart Object API has analogous relationships corresponding to it's SmartObject, Description, ObservableProperty, and the PropertyOfInterest classes.



The SmartObject class defines a VirtualEntity, ObservableProperty is an Attribute, PropertyOfInterest is a ValueContainer, Value is Instance of a Type, Descriptions are Metadata making up Service, Resource, and Device Descriptions.

Deployment Patterns

The current deployment pattern for the IoT can be approximately characterized by the channel model described in figure 23 of the IoT ARM document:

The Smart Object API supports the concept of a Smart Gateway, which acts as a Smart Object proxy for devices on the Constrained Network, adding semantic descriptors and providing a service interface for the device representation on the Web.

Other Smart Object IoT endpoints include smart sensors, which have enough resources to interact using the service interface, and user client devices such as smartphones, which also have sufficient resources to interact using http and REST interfaces. Smartphones can also act as ad-hoc gateways to connect to sensors over Bluetooth, USB, etc. in order to update Smart Object representations on the web.

The CoAP protocol is a good example of a constrained protocol that is easy to integrate using the Smart Object API. CoAP defines a RESTful interface and structured resources using URIs. A Smart Object instance can be created on the gateway to act as a proxy for the CoAP resources, and a simple CoAP agent can interact with sensors on the constrained network to update the Smart Object properties and it's web app subscriptions.

There are some fixed-function constrained network gateways that support simple web interfaces. These can be easily extended using a service proxy running on a small computer like a $60 Android stick or $35 Raspberry Pi. The service proxy can interact with the gateway's fixed web interface and provide Smart Object wrappers for the attached sensors. It can provide a local service environment for application agents and direct user device interaction.

Finally, there are smart sensors with e.g. Wi-Fi interfaces that can interact directly on the Internet. These might also be though of as sensors with integrated Smart Gateways. An Arduino based sensor with a serial Wi-Fi module has enough resources to interact using a proper subset of the Smart Object API, enough to interact directly with a Smart Object proxy on a web service or another gateway.

Integration with other Stacks and Standards

The Smart Object API is semantically identical both internally (program API) and externally (web API). This allows the API to be used as a library for web access to Smart Object instances, or by creating a local Smart Object instance that the application agent can synchronize sensor data to using subscriptions and data push.

The Agent can be used as a set of methods and handlers in a monolithic application program, or be deployed inside a service as an autonomous software entity that evaluates rules or algorithms in a data flow graph, synchronized by data updates from sensors, etc.

Standard web formats will be used for the web interface documents. HTML, XML, JSON, and RDF are the underlying types with PropertyOfInterest data type being negotiated semantically.

The data model should be interoperable with the Semantic Web linked data formats. GET on Smart Object level URIs should by default point to the RDF describing the Smart Object to enable further interaction.

The data model is fundamentally interoperable with CoAP, and allows simple transparent caching proxy implementation between CoAP and Smart Object API, using a CoAP Agent to manage the caching, subscription updates, etc. An instance of a Smart Object API can be created on a CoAP node if resources permit, by routing resources using URI-Path as per the CoAP Feature Analysis document.

Smart Object Service Framework

Some additional functions are needed to create a build-out infrastructure for a prototype Smart Object Service.

The Smart Object itself can have some common patterns built in:

Methods for self-assembling Smart Objects from Descriptions. Structural RDF elements describe ObservableProperty instances and Agent Instances. These instances should be created from the RDF itself, allowing Smart Objects to create and replicate themselves from sensor and composite object descriptions.

A semantic discovery and linkage engine is needed to allow application agents to connect to ObservableProperty instances of other Smart Objects. Facilitates examining RDF descriptions and constructing URIs to interact with and create subscriptions to properties of interest.

Reference agents for common patterns, e.g. timer, PID controller. Usable as templates for more sophisticated agents. Pluggable agents.

Services need a few additional features for managing collections of Smart Objects:

RDF crawler, database, indexing and discovery engine with SPARQL interface. A service can crawl the RDF and build search indices for first order discovery. Services can be built for large-scale discovery and linkage by crawling other services and aggregating results using MapReduce.

API capability-by-method key management framework. Key generator API using user secret. Key lifetime management.

URI router for path based routing to Smart Objects on service. Threaded web server interface. Facilitates scale-up of service to web volumes.

Summary

The Smart Object API is a Semantic Web Linked Data application for the Internet of Things (IoT). It consists of a URI-Object encapsulation of semantic and real-time data properties associated with features of interest.

The Smart Object architecture roughly conforms to the Virtual Entity, the Information Model, and the Channel Model set out in the IoT-A Architecture Reference Model (IoT-A ARM).

Supports direct interaction between smart sensors, smart gateways, cloud/internet services, and user devices. Interaction uses standard web protocols and formats and is semantically a superset of the CoAP protocol.

Service framework is to include object creation from semantic metadata, semantic database, discovery, and linkage, API capability keys, and threaded server.

Bibliography

SPITFIRE: Towards a Semantic Web of Things
Pfisterer, Roemer, et. al.
https://www.iti.uni-luebeck.de/fileadmin/sensornw/paper/IEEEComMag.pdf

A Resource Oriented Architecture for the Web of Things
Guinard, Trifa, Wilde
http://www.sciweavers.org/publications/resource-oriented-architecture-web-things

Building Blocks of the Internet of Things: State of the Art and Beyond
Serbanati, Medaglia, Ceipidor
http://cdn.intechopen.com/pdfs/17872/InTech-Building_blocks_of_the_internet_of_things_state_of_the_art_and_beyond.pdf

Linked Data - Design Issues
Tim Berners-Lee
http://www.w3.org/DesignIssues/LinkedData.html

The Scale-free nature of the Web
Tim Berners-Lee
http://www.w3.org/DesignIssues/Fractal.html

CoAP Feature Analysis
draft-shelby-6lowapp-coap-00
Shelby, et. al.
http://tools.ietf.org/html/draft-shelby-6lowapp-coap-00

Constrained Application Protocol (CoAP)
draft-ietf-core-coap-11
Shelby, et. al.
http://tools.ietf.org/html/draft-ietf-core-coap-11

Architectural Reference Model for the IoT (updated)
Updated reference model for IoT v1.5, Internet-of-Things Architecture EC Project
Bassi, Giacomin reviewers
D1 3_Architectural_Reference_Model_updated.pdf
http://www.iot-a.eu/public/public-documents/documents-1/1/1/copy_of_d1.2/at_download/file