Smart Objects and Semantic Linkage

Toward a prototype Smart Object

Michael J Koster
August 17, 2012

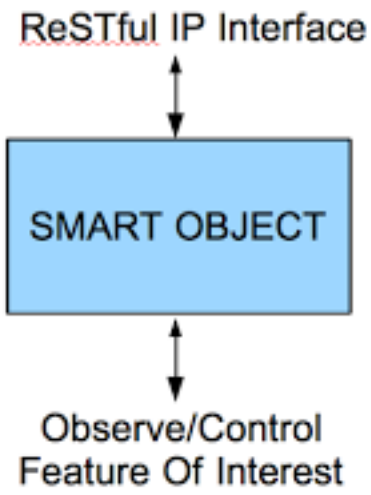
This is the second post in a series documenting my investigation into data models for pervasive interoperability on the Internet of Things. Here I'm looking into the information architecture inside the data model.

Previously, I mentioned that many sensors and devices already have the ability to connect to the internet. There is emerging a common way for devices and sensors to present themselves on the internet, which is through a RESTful interface. In short, Representational State Transfer is a style of communication that allows interfaces to be stateless and self-describing. Using REST allows two entities to exchange information is a well defined way without them having to know in advance what the other endpoint is going to do internally. The result is a standard and stable API that can be widely shared.

Smart Objects

Within the internet of things there is emerging the concept of a Smart Object. This is often described as a sensor or actuator which interacts using a REST interface on a network or on the internet.

Here is a somewhat more general concept of a smart object. In this document I'm going to use the term "Smart Object" to refer to a hardware device or software entity, pointed to by a URL, that implements the data models for the Semantic Web of Things. This is a logical broadening of the existing definition of a smart object from a component of a sensor network to a data node in the Semantic Web of Things.



A Smart Object observes or controls some feature of interest and interacts using a RESTful interface on a network such as the internet. The feature of interest can be practically any information source as long as there is a well known representation. The feature of interest may be a mashup of observable properties of other Smart Objects.

The Smart Object is a resource pointed to by a URL. It can be a smart sensor, a sensor observation or actuator control on a smarthome gateway, a resource on a cloud service that provides an information stream, or a device that informs the user e.g. an "orb". It can also be a data connector to a system that uses it's own proprietary data model. Practically anything from documents to real time data streams can be made into Smart Objects. Smart Objects can have arbitrary internal processes that filter, aggregate, or mashup data streams, process documents, or drive user controls.

There is not a current strong use-case for the extended functionality of the Smart Object interface, in that no large scale Internet of Things applications exist today. Common themes in the vision for the Internet of Things are location awareness, real time data consistency, granular scale (scale-less interactions), information mash-up, semantic discovery and linkage, self-aware systems.

The well known RESTful API operations of GET/SET/CREATE/DELETE for resources have been extended in CoAP to support data push using a subscribe/publish model. I believe publish-on-change data push is a very important mode in order to support both high volume and granular scale while maintaining real time data consistency.

I propose to add both semantic metadata operations and a push/subscription data transfer mode to the familiar REST interface for smart objects. I believe this can be the basis of a scalable, effective data model for a semantic web of Smart Objects.

Semantic Linkage of Smart Objects - A Semantic Web of Things

Smart Objects can be considered data sources. The capability of Smart Objects can be extended by providing an interface allowing the smart object to describe itself. A user, service, or agent can ask the Smart Object to describe itself and thus obtain a catalog of the observable properties of the object and their semantic descriptions.

This semantic metadata interface allows a crawler or application composer tool to discover the observable properties and other resources of a Smart Object, thus facilitating the chaining, aggregation, mash-up, etc. of the observable properties and resources from a specific collection of Smart Objects.

Data Push Capability - Data driven compute model

Data push is an optional method to propagate updates of observable properties or actuator states between Smart Objects. When a qualifying event such as a significant change in an observable property occurs, the information will be pushed, literally multicast, to other Smart Objects.

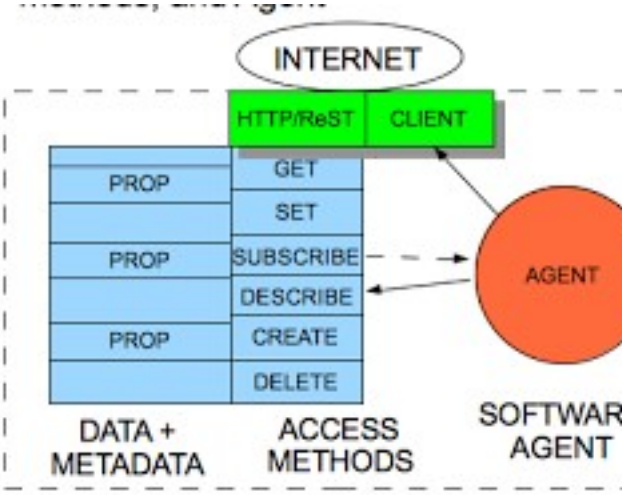
Smart Objects need the capability to respond to changes pushed to their input properties without having to arbitrarily poll resources. This requires some kind of callback to a handler or agent in response. The Smart Object proposed contains agent code and a framework to link handlers to property updates pushed from other Smart Objects.

A data driven, semantically linked web of things

The combination of semantic capability and data push allows Smart Objects to be assembled into a semantically linked data flow graph to construct larger systems. Smart Objects can be instantiated and connected to data sources, and outputs routed to other smart objects for filtering, aggregation, display, etc.

A Prototype Smart Object

A simple prototype can be constructed to create a testbed for the Smart Object and the Semantic Web of Things. Here is a simplified block diagram of the components for a prototype Smart Object.



Smart Object

Data and metadata (semantic tags) are accessed by a set of RESTful methods which are both mapped onto the http interface and available to a local software agent.

The software agent contains the application software such as timers, PID controllers, filters, aggregators and the callback handlers for data push processing. The software agent is also responsible for sensor and actuator control in Smart Sensors, user interface connections, and connectors to other data models e.g. legacy databases.

Semantic metadata are stored in the form of RDF triples that describe the resources and observable properties of the Smart Object. CREATE, DELETE, SET, and GET should be able to operate on the metadata, and the DESCRIBE method is proposed to point to a resource containing the full RDF description of the object or one of it's observable properties.

Smart Object Registry for semantic discovery and linkage

A Smart Object can also function as a registry for itself and other Smart Objects, providing a resource in the network that can support discovery and linkage through a well known URL. Registry objects would provide additional RDF query functions, e.g. SPARQL interface and index to the triple store, to support semantic discovery and linking.

Crawler software on the registry nodes can obtain the RDF from Smart Objects using the DESCRIBE method, and populate and index a triplestore. A SPARQL query engine on the triplestore indices returns URLs of Smart Objects that contain observable properties or resources that satisfy the query predicates. Any additional semantic metadata necessary to perform a particular mash-up can be obtained from the registry that describes the selected objects and resources. A new Smart Object is created that has as it's inputs properties of other Smart Objects, and as it's output the new composite properties resulting from the mash-up

algorithm. If real time push updates of observable properties is desired, subscriptions are created to the Smart Objects that are the data sources.

Data models

The data models for interoperability consist of the semantic definitions of the Smart Object REST interface including the push and metadata operations, the use of embedded RDF to describe the observable properties and resources of a Smart Object, and the Smart Object registry interface for semantic discovery and linking.

The choice of data types, representations, sizes, formats, etc. is left entirely to the domain of interest to use the applicable ontologies. The choice of object granularity vs. complexity is entirely up to the system designer and information architect.

The location of registries and system of well known registry paths in larger deployments is as yet unspecified and the subject of investigation. It's expected that some design patterns will emerge and some conventions and protocols refined as a result.

Related Work

The SPITFIRE project associates semantic metadata with RESTful smart sensors, and provides a triplestore with SPARQL interface for semantic discovery and linkage.

https://www.iti.uni-luebeck.de/fileadmin/sensornw/paper/IEEEComMag.pdf

CoAP is a constrained application protocol meant to use CoRE, a REST interface for constrained environments. It's emerging in the set of 6LoWPAN related standards. The interesting thing about CoAP is the inclusion of a subscription/publish-on-change mode.

Here's a concise feature analysis of CoAP:

https://datatracker.ietf.org/doc/draft-shelby-core-coap/

Next Up

I'm currently looking at tools for manually building and maintaining registry information and beginning implementation of the prototype.