I. INTRODUCTION

Standards for constrained devices are rapidly consolidating and the availability of IP on constrained devices enabled these devices to easily connect to the Internet. The IETF has also created a set of specifications for such IP-enabled devices to work in a Web-like fashion. One such protocol is the Constrained Application Protocol (CoAP) [1] that provides request/response methods, ways to identify resources, discovery mechanisms, etc. similar to HTTP [2] but for use in constrained environments.

However, the use of standardized protocols does not ensure interoperability on the application layer. Therefore, there is a clear need for being able to communicate using structured data models on top of protocols like CoAP and HTTP.

IPSO Smart Objects provide a common design pattern, an object model, to provide high level interoperability between Smart Object devices and connected software applications on other devices and services. IPSO Objects are defined in such a way that they do not depend on the use of CoAP, any RESTful protocol is sufficient. Nevertheless, to develop a complete and interoperable solution the Object model is based on the Open Mobile Alliance Lightweight Specification (OMA LWM2M) [3], which is a set of management interfaces built on top of CoAP in order to enable device management operations (bootstrapping, firmware updates, error reporting, etc.). While LWM2M uses objects with fixed mandatory resources, IPSO Smart Objects use a more reusable design.

II. DATA MODEL

The data model for IPSO Smart Objects consists of four parts:
1) Object Representation
2) Data Types
3) Operations
4) Content Formats

A. Object Representation

Objects and resources are implicitly mapped into the URI path hierarchy by following the OMA LWM2M object model, in which each URI path component sequentially represents the Object Type ID, the Object Instance ID and the Resource Type ID. More precisely the structure consists of three unsigned 16-bit integers separated by the character '/' in the following form: Object ID/Instance ID/Resource ID. This URI template approach follows the Web Linking [4] and the IETF CoRE Link Format [5]. Objects are typed containers, which define the semantic type of instances. Instances represent specific object types at runtime, and allow Smart Object endpoints to expose multiple sensors and actuators of a particular type. Object instances are themselves containers for resources, which are the observable properties of an object.

Figure 1 shows an example URI of a temperature sensor.

Fig. 1. Temperature Sensor URI Example.

Semantically, the object type represents a single measurement, actuation, or control point for example a temperature sensor, a light (actuator), or an on-off switch (control point).

A resource specifies a particular view or active property of an object. For example, a temperature sensor object might expose the current value (most recent reading), also the minimum and maximum possible reading, the minimum and maximum reading in an interval, and attributes like engineering units and application type.
Attributes describe the metadata configuration, settings, and state of an object or resource, and are discoverable by reading the link-format data of an object or resource. Multiple attributes may be serialized in the link-format descriptors that an object exposes. Some attributes are immutable for a given object or resource type. For example, the static read, write, and execute capability attributes are derived from a Smart Object’s definition file, while other attributes, like the LWM2M Notification Attributes, are used to dynamically configure a particular object instance or resource. Attributes are represented using the IETF CoRE Link Format (RFC 6690) or an equivalent mapping to other content formats, for example, application/json+Id.

This abstraction allows application software to use simple APIs. For complex objects, linking of an object to another object through an object link resource is allowed. This enables the recursion to be handled at the object level, using design patterns similar to web linking. An application client can consume a devices API without knowing its structure and attributes a priori.

B. Data Types
IPSO Smart Objects re-use the data types defined in the OMA LWM2M specification [3].
1) String: A UTF-8 string
2) Integer: An 8, 16, 32 or 64-bit signed integer.
3) Float: A 32 or 64-bit floating point value.
4) Boolean
5) Opaque: A sequence of binary octets.
6) Time: Unix Time.
7) Object Link: The object link is used to refer an instance of a given object.

C. Operations
IPSO Objects and their resources have the same operations as their counterparts in the OMA LWM2M specification [3] with the same semantics.
1) Resource values: Read, Write, Execute (restricted by the Access Type field)
2) Object Instances: Create, Delete (restricted by the Multiple Instances field)
3) Objects and their instances: Read, Write
4) Attributes: Set, Discover

D. Content Formats
Content formats are those specified by the OMA LWM2M specification [3]:
1) Resource values: text/plain, tlv
2) Objects: text/senml+json, application/cbor, binary/tlv
3) Attributes: link-format, link-format+json

III. HUMIDITY SENSOR EXAMPLE
Specification authors use different ways to describe resources exposed by IoT devices. Often natural text is used and sometimes more formal techniques are relied on. For the definition of the IPSO Smart Objects tables with natural language descriptions and XML (to offer machine-readable descriptions) are used.

The following is an example of a humidity sensor that contains the sensor value, units, min and max measured values, min and max range values and a resets for those. Figure 2 shows the object and resource definitions in a tabular form and the definition in XML is shown in Appendix A.

IV. COMPOSITE OBJECTS
As devices increase in complexity (e.g., from a sensor to an appliance, from a switch to a complex actuator) the need to link resources to create more complex objects or "Composite Objects" arises. Such a composite object can, for example, be constructed with a single reusable type "generic composite object" with one ID. The resources may be of a generic reusable link type, also using a single ID, with multiple instances allowed. For example, '4000/0/6700/0' where 4000 is a "composite object" and 6700 is "generic object link". Composite objects offer higher granularity than one large nested object would. An observer of a device represented as a composite object could reduce bandwidth utilization by observing only the linked object instances instead of the full object. Figure 2 shows an example, performing a GET operation to the IPSO thermostat composite object "/3300/0/7100" would retrieve an object link to "/3300/0/7100/0".
V. IPSO SMART OBJECTS

A. Starter Pack

This IPSO Smart Objects Starter Pack describes 18 Smart Objects, including a temperature sensor, a light controller, an accelerometer, a presence sensor, and other types of sensors and actuators. These objects are common in many application domains. Appendix B shows the list of objects defined in the Starter Pack.

With this initial publication the IPSO Alliance aimed to offer developers and standardization experts a starting point from which to build more complex objects in order to address vertical IoT market segments. One important design criteria in the design of the IPSO Smart Objects was and is to making it easy for developers to derive new objects based on their use case needs, while promoting interoperability to an extent as is practical. Naturally, a device that has not seen a newly defined object cannot know the semantics even if the contained resources can be understood on a syntactical level. However, re-use of existing object and resource definitions allows application developers to re-use code.

B. Expansion Pack

To complement the initial set of objects, the IPSO Smart Object Expansion Pack was published. The Expansion Pack adds 16 common template sensors, 6 special template sensors, 5 actuators and 6 control switch types.

Some of the newly defined objects are generic in nature, such as voltage, altitude or percentage, while others are more specialized like the Color Object or the Gyrometer Object. New actuators and controllers are defined such as timer, buzzer, joystick and level. All of these objects were found to be necessary on a variety of use case domains.

Appendix C lists the objects defined in the IPSO Expansion Pack.
C. Extensibility

Apart from the objects published by the IPSO Alliance developers and standardization experts are encouraged to define new objects tailored to their use cases if the already available functionality is insufficient. The Starter and Expansion Pack provide basic examples for common sensors and actuators. Developers may extend the existing object set and submit them to IPSO.

VI. ACKNOWLEDGMENTS

The authors would like to thank Bill Silverajan and Jens Eliasson for their feedback.

REFERENCES


APPENDIX

APPENDIX A: HUMIDITY OBJECT DEFINITION IN XML FORMAT

The following is the definition document for the Humidity Object in XML.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LWM2M>
  <Object ObjectType="MODefinition">
    <Name>Humidity</Name>
    <Description>This IPSO object should be used with a humidity sensor to report a humidity measurement. It also provides resources for minimum/maximum measured values and the minimum/maximum range that can be measured by the humidity sensor. An example measurement unit is relative humidity as a percentage (ucum:%).
    </Description>
    <ObjectID>3304</ObjectID>
    <ObjectURN>urn:oma:lwm2m:ext:3304</ObjectURN>
    <MultipleInstances>Multiple</MultipleInstances>
    <Mandatory>Optional</Mandatory>
    <Resources>
      <Item ID="5700">
        <Name>Sensor Value</Name>
        <Operations>R</Operations>
        <MultipleInstances>Single</MultipleInstances>
        <Mandatory>Mandatory</Mandatory>
        <Type>Float</Type>
        <RangeEnumeration></RangeEnumeration>
        <Units>Defined by "Units" resource.</Units>
        <Description>Last or Current Measured Value from the Sensor</Description>
      </Item>
      <Item ID="5601">
        <Name>Min Measured Value</Name>
        <Operations>R</Operations>
        <MultipleInstances>Single</MultipleInstances>
        <Mandatory>Optional</Mandatory>
        <Type>Float</Type>
        <RangeEnumeration></RangeEnumeration>
        <Units>Defined by "Units" resource.</Units>
        <Description>Minimum Measured Value from the Sensor</Description>
      </Item>
      <Item ID="5501">
        <Name>Max Measured Value</Name>
        <Operations>R</Operations>
        <MultipleInstances>Single</MultipleInstances>
        <Mandatory>Optional</Mandatory>
        <Type>Float</Type>
        <RangeEnumeration></RangeEnumeration>
        <Units>Defined by "Units" resource.</Units>
        <Description>Maximum Measured Value from the Sensor</Description>
      </Item>
    </Resources>
  </Object>
</LWM2M>
```
<RangeEnumeration></RangeEnumeration>
<Units>Defined by "Units" resource.</Units>
<Description>The minimum value measured by the sensor since power ON or reset</Description>
</Item>
<Item ID="5602">
  <Name>Max Measured Value</Name>
  <Operations>R</Operations>
  <MultipleInstances>Single</MultipleInstances>
  <Mandatory>Optional</Mandatory>
  <Type>Float</Type>
  <RangeEnumeration></RangeEnumeration>
  <Units>Defined by "Units" resource.</Units>
  <Description>The maximum value measured by the sensor since power ON or reset</Description>
</Item>
<Item ID="5603">
  <Name>Min Range Value</Name>
  <Operations>R</Operations>
  <MultipleInstances>Single</MultipleInstances>
  <Mandatory>Optional</Mandatory>
  <Type>String</Type>
  <RangeEnumeration></RangeEnumeration>
  <Units>Defined by "Units" resource.</Units>
  <Description>The minimum value that can be measured by the sensor</Description>
</Item>
<Item ID="5604">
  <Name>Max Range Value</Name>
  <Operations>R</Operations>
  <MultipleInstances>Single</MultipleInstances>
  <Mandatory>Optional</Mandatory>
  <Type>Float</Type>
  <RangeEnumeration></RangeEnumeration>
  <Units>Defined by "Units" resource.</Units>
  <Description>The maximum value that can be measured by the sensor</Description>
</Item>
<Item ID="5701">
  <Name>Sensor Units</Name>
  <Operations>R</Operations>
  <MultipleInstances>Single</MultipleInstances>
  <Mandatory>Optional</Mandatory>
  <Type>String</Type>
  <RangeEnumeration></RangeEnumeration>
  <Units></Units>
  <Description>Measurement Units Definition e.g. "Cel" for Temperature in Celsius.</Description>
</Item>
<Item ID="5605">
  <Name>Reset Min and Max Measured Values</Name>
  <Operations>E</Operations>
  <MultipleInstances>Single</MultipleInstances>
<Mandatory>Optional</Mandatory>
<Type>Opaque</Type>
<RangeEnumeration></RangeEnumeration>
<Units></Units>
<Description>Reset the Min and Max Measured Values to Current Value</Description>
</Item>
</Resources>
<Description2></Description2>
</Object>
</LWM2M>

**APPENDIX B: IPSO STARTER PACK**
The IPSO Starter Pack defines the objects shown in Table I.

<table>
<thead>
<tr>
<th>Object</th>
<th>Object ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital</td>
<td>3200</td>
</tr>
<tr>
<td>Digital Output</td>
<td>3201</td>
</tr>
<tr>
<td>Analogue Input</td>
<td>3202</td>
</tr>
<tr>
<td>Analogue Output</td>
<td>3203</td>
</tr>
<tr>
<td>Generic Sensor</td>
<td>3300</td>
</tr>
<tr>
<td>Illuminance Sensor</td>
<td>3301</td>
</tr>
<tr>
<td>Presence Sensor</td>
<td>3302</td>
</tr>
<tr>
<td>Temperature Sensor</td>
<td>3303</td>
</tr>
<tr>
<td>Humidity Sensor</td>
<td>3304</td>
</tr>
<tr>
<td>Power Measurement</td>
<td>3305</td>
</tr>
<tr>
<td>Actuation</td>
<td>3306</td>
</tr>
<tr>
<td>Set Point</td>
<td>3308</td>
</tr>
<tr>
<td>Load Control</td>
<td>3310</td>
</tr>
<tr>
<td>Light Control</td>
<td>3311</td>
</tr>
<tr>
<td>Power Control</td>
<td>3312</td>
</tr>
<tr>
<td>Accelerometer</td>
<td>3313</td>
</tr>
<tr>
<td>Magnetometer</td>
<td>3314</td>
</tr>
<tr>
<td>Barometer</td>
<td>3315</td>
</tr>
</tbody>
</table>

**APPENDIX C: IPSO EXPANSION PACK**
The IPSO Expansion Pack defines the objects shown in Table II.
## TABLE II
IPSÖ EXPANSION PACK.

<table>
<thead>
<tr>
<th>Object</th>
<th>Object ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>3316</td>
</tr>
<tr>
<td>Current</td>
<td>3317</td>
</tr>
<tr>
<td>Frequency</td>
<td>3318</td>
</tr>
<tr>
<td>Depth</td>
<td>3319</td>
</tr>
<tr>
<td>Percentage</td>
<td>3320</td>
</tr>
<tr>
<td>Altitude</td>
<td>3321</td>
</tr>
<tr>
<td>Load</td>
<td>3322</td>
</tr>
<tr>
<td>Pressure</td>
<td>3323</td>
</tr>
<tr>
<td>Loudness</td>
<td>3324</td>
</tr>
<tr>
<td>Concentration</td>
<td>3325</td>
</tr>
<tr>
<td>Acidity</td>
<td>3326</td>
</tr>
<tr>
<td>Conductivity</td>
<td>3327</td>
</tr>
<tr>
<td>Power</td>
<td>3328</td>
</tr>
<tr>
<td>Power Factor</td>
<td>3329</td>
</tr>
<tr>
<td>Rate</td>
<td>3346</td>
</tr>
<tr>
<td>Distance</td>
<td>3330</td>
</tr>
<tr>
<td>Energy</td>
<td>3331</td>
</tr>
<tr>
<td>Direction</td>
<td>3332</td>
</tr>
<tr>
<td>Time</td>
<td>3333</td>
</tr>
<tr>
<td>Gyrometer</td>
<td>3334</td>
</tr>
<tr>
<td>Color</td>
<td>3335</td>
</tr>
<tr>
<td>GPS Location</td>
<td>3336</td>
</tr>
<tr>
<td>Positioner</td>
<td>3337</td>
</tr>
<tr>
<td>Buzzer</td>
<td>3338</td>
</tr>
<tr>
<td>Audio Clip</td>
<td>3339</td>
</tr>
<tr>
<td>Timer</td>
<td>3340</td>
</tr>
<tr>
<td>Addressable Text Display</td>
<td>3341</td>
</tr>
<tr>
<td>On/Off Switch</td>
<td>3342</td>
</tr>
<tr>
<td>Push Button</td>
<td>3347</td>
</tr>
<tr>
<td>Level Control</td>
<td>3343</td>
</tr>
<tr>
<td>Up/Down Control</td>
<td>3344</td>
</tr>
<tr>
<td>Multistate Selector</td>
<td>3348</td>
</tr>
<tr>
<td>Multiple Axis Joystick</td>
<td>3345</td>
</tr>
</tbody>
</table>