What problems are we trying to solve?

Hannes Tschofenig

Possible Goals

1. **End users** need descriptions of resources and actions in their language of choice for configuring if/then style rules.
2. **Administrators** need descriptions of resources and actions for configuring security policies.
3. **Standards orgs** need data models for review, publication, and potentially compliance testing / certification.
4. **Developers** need data models for manual or automated code generation
5. **Debugging tools** want them for generic object browsers, etc.
6. **Translators** want them for dynamic mapping to other protocols without a priori knowledge
7. **Software** needs info for discovery of meta-data and for driving interactions.
8. … others?

Many different solutions today for getting data models, with varying tradeoffs

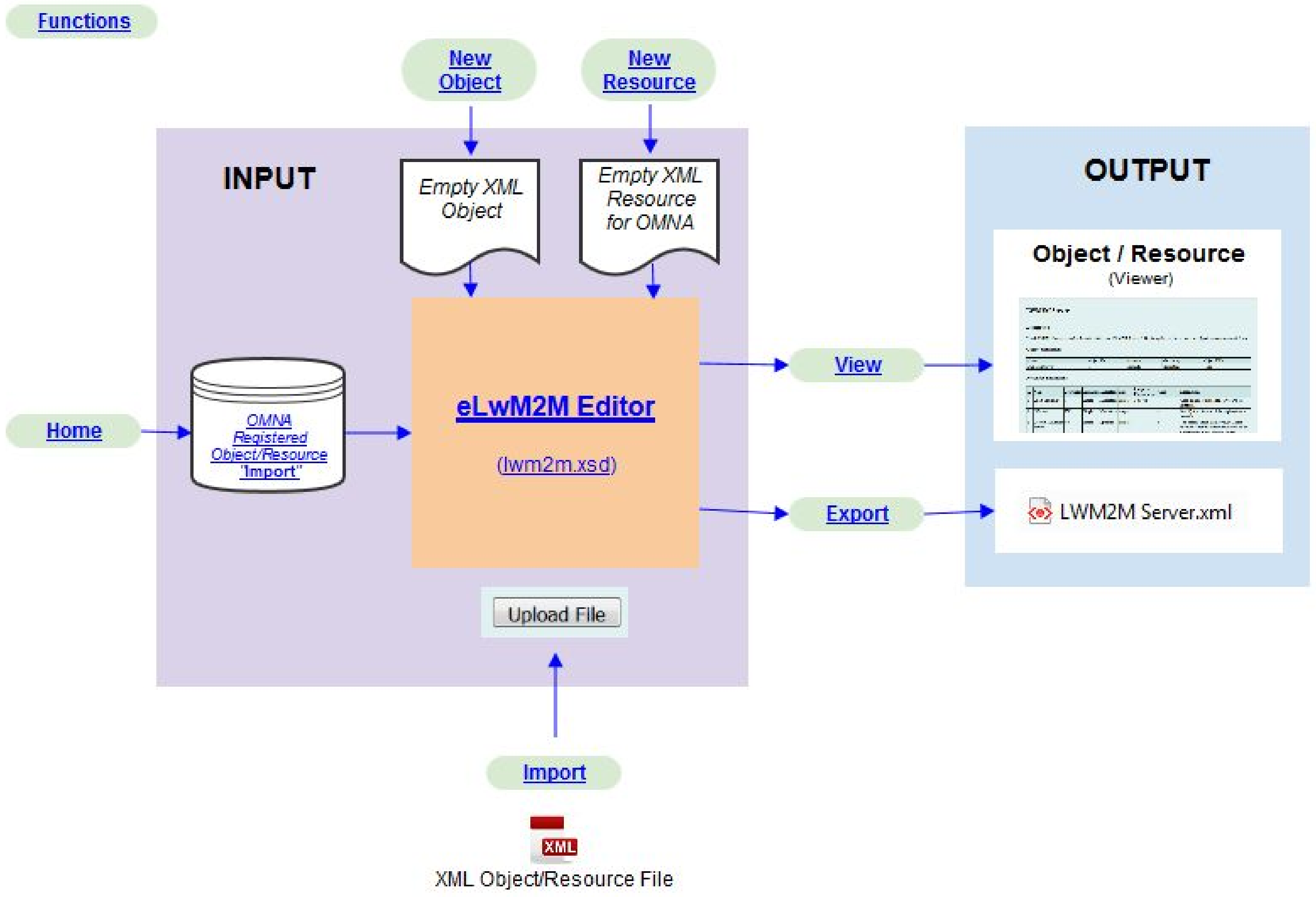
- What form do you get it in:
  - Extracted from specification, or obtained directly in data model form?
- Where do you get it from:
  - A cloud repository? The vendor’s site? A device itself?
- Does it all come in one piece or are there different pieces possibly from different places?
  - E.g., syntax vs end-user descriptions in language X vs developer-specific comments

Example: OMA & IPSO

OMA: LWM2M Schema

- Schema was created to support creation and handling Objects & Resources.
- Schema can be found at: http://technical.openmobilealliance.org/tech/profiles/LWM2M.xsd
- This schema is NOT part of the OMA LightweightM2M v1.0 release, (LwM2M)
- The new LwM2M Objects and Resources editor, (eLwM2M Editor), is on trials. The plan is to release it at the end of March 16.
- Note: The schema needs an update as it still contains some problems.

OMA LwM2M Editor



OMA LwM2M Editor

- This tool was created to assist OMA Working Groups and other non-OMA groups to create and handling LwM2M Objects/Resources
- Tool output: xml file and table describing the Object/Resources are aligned

The Editor will:

- Create new Objects/Resources by input information into a table,
- Produce the xml file based on the content of the table,
- Allows to copy and past the newly created table into a OMA template or Word Document
  - This simplify the creation of new Object/Resources documents
- Download the newly created xml file
  - Well-formed and validated according to the lwm2m.xsd schema
- Import xml files into the tool
- Check if the xml file is well-formed and valid as per lwm2m.xsd schema
- Import OMA Object/Resources (registered with OMNA, OMA Naming Authority)

Reference Material

The new editor tools will be publically available at:

For OMA LightweightM2M:

- http://devtoolkit.openmobilealliance.org/OEditor/

- For OMA Device Management:

- http://devtoolkit.openmobilealliance.org/DMEditor/

For feedback and source code:

- https://github.com/OpenMobileAlliance/OMA-Objects-Resources-Editor

IPSO

- Uses LWM2M Object Model.
- Reusable Object IDs and Resource IDs.
- Makes use of the OMA LWM2M schema for object/resource registrations to fulfill the requirements by OMA.
- Protocol Independent (CoAP, LWM2M, MQTT, HTTP…) if support addressing, content formats and data types.
- Encoding Independent (JSON, TLV, SenML…)

Humidity Sensor Example: Table Representation

Object	Object ID	Object URN	Multiple Instances?	Description
IPSO Humidity	3304	**urn:oma:lwm2m:ext:3304**	Yes	Relative humidity sensor, example units = %

Resource Name	Resource ID	Access Type	Multiple Instances?	Mandatory	Type	Range or Enumeration	Units	Descriptions
Sensor Value	5700	R	No	Mandatory	Float			Last or Current Measured Value from the Sensor
Units	5701	R	No	Optional	String			Measurement Units Definition e.g. “Cel” for Temperature in Celsius.
Min Measured Value	5601	R	No	Optional	Float	Same as Measured Value	Same as Measured Value	The minimum value measured by the sensor since power ON or reset
Max Measured Value	5602	R	No	Optional	Float	Same as Measured Value	Same as Measured Value	The maximum value measured by the sensor since power ON or reset
Min Range Value	5603	R	No	Optional	Float	Same as Measured Value	Same as Measured Value	The minimum value that can be measured by the sensor
Max Range Value	5604	R	No	Optional	Float	Same as Measured Value	Same as Measured Value	The maximum value that can be measured by the sensor
Reset Min and Max Measured Values	5605	E	No	Optional	Opaque			Reset the Min and Max Measured Values to Current Value

XML representation available in position paper: https://www.iab.org/wp-content/IAB-uploads/2016/03/ipso-paper.pdf

Example: SenML

SenML - Too **Little** Meta Data

Consider

```
[
      { "bn": "urn:dev:ow:10e2073a01080063" },
      { "t": 1276020076, "v":23.5, "u":"Cel" },
      { "t": 1276020091, "v":23.6, "u":"Cel" }
]
```

We know the globally unique name of the sensor is `urn:dev:ow:10e2073a01080063` and it is a temperature of `23.5` degrees Celsius at time `1276020076`

We don’t know the device type, OS version, manufacturer, which data model it uses or the access control lists. Yes, you might need that for some management but you don’t need it for most use cases for a sensor or simple actuator

Goal is to cut it down to the bare minimum that still accomplishes many (but not all) of the use cases and is really simple to understand and use

The web, like IP, is successful because it started simple and easy

SenML - Too **Much** Meta Data

One might say:

> The name of the sensor and unit is duplicated meta data so why send it in every measurement? Get rid of the meta data and just send 23.5

Keeping the name and time allows the data to be stored in a schemaless DB and still processed. It allows many cache, aggregation, and filters to be applied.

It can increase performance for servers receiving millions of measurements to be able to handle the measurement in a stateless way

SenML tries to balance the meta data to make it easy for small simple devices with limited connectivity while being easy for large servers using current big data style tools

Example: HATEOAS

HATEOAS / Hypermedia-driven Applications

- Allow systems to evolve independently to handle change
  - Replace things without (manual) reconfiguration
  - Change the control flow during runtime
  - Add new devices and services
- Drive the application at runtime (“Engine of Application State”)
  - Atomic interaction steps (request-response exchanges)
  - In-band descriptions (links, forms, and relation types as semantic annotations)

➔ Self-describing interaction model

Example: AllJoyn

Tooling

- AllJoyn uses data model formats in XML with an XSD, based on D-Bus
- Interface Review Board publishes guidelines for data models and reviews data models against the guidelines
- **AJXmlCop** is an open source tool to automatically check a data model against many of the published guidelines
- **GetAJXml** is a tool to fetch the data model from a device
- **AllJoyn Explorer** is a generic object browser that consumes data models at runtime
- Two independent code generators exist, for different programming languages & platforms

Overview of authoring mechanics

1. Create introspection XML using an XML editing tool that validates against the XSD
2. Run **ajxmlcop** to check it against IRB design guidelines
3. Generate code
4. Submit for IRB review via Gerrit and update based on feedback
    - Required for standard interfaces, but optional for proprietary

Example submission (simplified)

```xml
<node xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://www.allseenalliance.org/schemas/introspect.xsd">
  <interface name="org.alljoyn.SmartSpaces.Operation.DishWashingCyclePhase">
    <description language="en">This interface provides a capability to monitor the cycle phase of the dishwasher.</description>
    <annotation name="org.alljoyn.Bus.Secure" value="true"/>
    <struct name="CyclePhaseDescriptor">
      <field name="phase" type="y"/>
      <field name="name" type="s"/>
      <field name="description" type="s"/>
    </struct>
    <property name="Version" type="q" access="read">
      <description language="en">Interface version</description>
      <annotation name="org.freedesktop.DBus.Property.EmitsChangedSignal" value="true"/>
    </property>
    <property name="CyclePhase" type="y" access="read">
      <description language="en">Current cycle phase</description>
      <annotation name="org.freedesktop.DBus.Property.EmitsChangedSignal" value="true"/>
    </property>
    <method name="GetVendorPhasesDescription">
      <description language="en">Get cycle phases description</description>
      <arg name="languageTag" type="s" direction="in">
        <description language="en">Preferred language to use in selecting output strings</description>
      </arg>
      <arg name="phasesDescription" type="[CyclePhaseDescriptor]" direction="out">
        <description language="en">Cycle phases description</description>
      </arg>
    </method>
  </interface>
</node>
```

Example: W3C Thing Description (TD)

W3C WoT TD

- Root descriptions in machine-understandable RDF
  - Semantic reasoning
  - Linking and automatic handling of vocabulary (e.g., ontology matching)
- Minimal vocabulary set to describe things
  - Capabilities
  - How to use
- Extensible with domain-specific and unspecific context
- Ability to retrofit existing (Web) APIs with TD

Example: OpenDOF

OpenDOF

Code generation, see

https://interface.opendof.org/interface-repository/interface?cmd=GetInterface&trans=html&repo=opendof&user=PSLCL&group=none&iid=dof/1.0225https://interface.opendof.org/interface-repository/interface?cmd=GetInterface&trans=html&repo=opendof&user=PSLCL&group=none&iid=dof/1.0225

Example: SAREF

SAREF

SAREF is an OWL-DL ontology with 124 classes, 56 object properties and 28 datatype properties

Documentation available at http://ontology.tno.nl/saref (now redirected to the permanent URL w3id.org/saref). The

ontology file can be downloaded at http://ontology.tno.nl/saref.ttl

Project website https://sites.google.com/site/smartappliancesproject

SAREF standardized by ETSI in “ETSI TS 103 264 V1.1.1 (2015-11)”

http://www.etsi.org/deliver/etsi_ts/103200_103299/103264/01.01.01_60/ts_103264v010101p.pdf

SAREF acknowledged as a first ontology standard in the current IoT standardization activities

https://ec.europa.eu/digital-agenda/en/blog/new-standard-smart-appliances-smart-home

http://ec.europa.eu/DocsRoom/documents/14681/attachments/1/translations/en/renditions/native

Summary

Possible Goals

1. **End users** need descriptions of resources and actions in their language of choice for configuring if/then style rules.
2. **Administrators** need descriptions of resources and actions for configuring security policies.
3. **Standards orgs** need data models for review, publication, and potentially compliance testing / certification.

OMA/IPSO, OCF, AllSeen, ...

1. **Developers** need data models for manual or automated code generation

Yang, OpenDOF

1. **Debugging tools** want them for generic object browsers, etc.

Bluetooth Low Energy Characteristic Descriptors

1. **Translators** want them for dynamic mapping to other protocols without a priori knowledge
2. **Devices** want them for discovery of meta-data and for driving interactions.

W3C Thing Descriptions, HATEOS, SenML