SMART DEVICE TEMPLATE V3

ANDREAS KRAFT, DT

HGI
YOUR GATEWAY TO THE SMART HOME
Goals

Describe devices and device services in a way which is independent of the LAN technology in a format which is convenient and reliable for integration.

1. **Keep it simple**, especially for manufacturers to contribute
2. **Modularity** for functions and device types
3. Make it easy for developers to create unified APIs
4. Be **independent** of underlying home-area network technologies

Make it available under an **open license**.
## Basic Components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domain</strong></td>
<td>Unique name, or &quot;wrapper&quot; which acts like a namespace, set by the organization creating the SDT, allowing reference to a package of definitions for the contained ModuleClasses and device definitions. Can be referenced when extending ModuleClasses. It has two possible uses: to select the scope of a technology domain, or to set the scope of a use case domain (like Home, SmartGrid, etc).</td>
</tr>
<tr>
<td><strong>Device &amp; Sub-Device</strong></td>
<td>Physical, addressable, identifiable appliance/sensor/actuator.</td>
</tr>
<tr>
<td><strong>ModuleClass</strong></td>
<td>Specification of a single service with one or more service methods, the involved abstracted data model and related events. The expectation is that each separate service which may be used in many kinds of Devices (like PowerON/OFF, Open/Close, ...) will be described by a ModuleClass which can be re-used in many Device definitions.</td>
</tr>
<tr>
<td><strong>Property</strong></td>
<td>Property elements are used to append to Devices and their ModuleClass elements with arbitrary additional information.</td>
</tr>
</tbody>
</table>
Main Idea: re-usable XML modules for describing most common functions

HGI offers a library of "module" elements, which conform to the ModuleClass, for the most common functions:

- A boolean actuator (like power switch)
- A real-value sensor (like temperature sensor)
- A boolean sensor (like window sensor)

Features, Attributes, Info about internal states, ....

Power Switch (for my music box)

Data Model (XML)

Description for Applications (and Programmers) → Data Model (XML)

Application developers can make use of a small set of common functional elements, not adapting API code for every HAN

HGI offers a library of “module” elements, which conform to the ModuleClass, for the most common functions:

- A boolean actuator (like power switch)
- A real-value sensor (like temperature sensor)
- A boolean sensor (like window sensor)

Features, Attributes, Info about internal states, ....

Power Switch (for my music box)

Data Model (XML)

Description for Applications (and Programmers) → Data Model (XML)

Application developers can make use of a small set of common functional elements, not adapting API code for every HAN

HGI offers a library of “module” elements, which conform to the ModuleClass, for the most common functions:

- A boolean actuator (like power switch)
- A real-value sensor (like temperature sensor)
- A boolean sensor (like window sensor)

Features, Attributes, Info about internal states, ....

Power Switch (for my music box)

Data Model (XML)

Description for Applications (and Programmers) → Data Model (XML)

Application developers can make use of a small set of common functional elements, not adapting API code for every HAN
How Things Could Come Together

- oneM2M / MAS
- Ontology
  - Smart Device Template
    - RURTIME: Describe Ontology Specifics
    - Repository & Tool Support (e.g. Eclipse VORTO)
  - Smart Device Template
    - PROCESS: Collect Technology Specifics
- Application SDK & Ontology Layer
- Abstraction Layer
  - Technology Binding
  - Technology Binding
  - Technology Binding

OSGi
  - Application Integration
  - Eclipse SmartHome
  - Device Connectivity
May 2015 - Version 1.0
Dezember 2015: public draft

• Basic concepts of the Smart Device Template
June 2015 - Version 2.0

- Device hierarchy
- More data types
- Better data points & device information
- Based on contributions from SAREF, ULE Alliance, Energy@Home
SDT Evolution

December 2015 - Version 3.0

- Complex data types
- Meta-data
- Property lists
SDT GitHub Repository

- All versions, tools, and contributions are available on GitHub.

- Under APL2 license.

https://homegateway/smartdevicetemplate
SDTTTool

- Convert from SDT2 to SDT3
- Generate documentation
  - Plain Text
  - Markdown
  - OPML (Mindmap)
- Generate Vorto import format (work-in-progress)
- Generate Java interfaces and classes

https://github.com/homegateway/sdttool
CONTRIBUTIONS & COLLABORATIONS
Information Models

**EchonetLite**

https://github.com/ECHONET-Consortium/ECHONET-SDT- Contribution

**ULE Alliance**

https://github.com/ULE-Alliance/HANFUN-SDT

**Energy@Home**

https://github.com/ismb/Energy-home-SDT- Contribution
Eclipse Vorto

- Eclipse-based tooling to describe device characteristics (Properties, Services) in form of Function Blocks
- An Information Model describes a device in its entirety composed of several Function Blocks
- Platform specific code generators can be used to transform information models in code artifacts
- Eclipse hosted Repository for Information Models in order to build community / ecosystem

http://www.eclipse.org/vorto
THE FUTURE
oneM2M – Modeling in TS-0023
Home Appliances Information Model and Mapping

• oneM2M is the best home for SDT, to allow extension as usage in oneM2M might require.
• SDT was selected in TR-0017 - Home Domain Abstract Information Model as the modeling scheme for oneM2M.
• SDT v3.0 is used in oneM2M’s Management, Abstraction and Semantics (MAS) Working Group to create and specify the Home Information modelling.
  – MAS will recommend oneM2M’s Steering Committee to take SDT under oneM2M governance.
• Future SDT hosting will be under ETSI
  – Using ETSI’s GitLab infrastructure
Role of Information Modeling for Interoperability

- Native Technology: AllJoyn, ECHONET, OIC, Zigbee, Others
- Information Model: Smart Meter, Lighting, Other Devices
- Generic Interworking
- oneM2M System

Interworking

Semantics based Common Ontology (ex. Device, Operation...)

Existing Resources

Device Abstraction
Information Modelling in oneM2M

- **Property**
  - country
  - deviceID
  - deviceType
  - deviceName
  - deviceModelName
  - ...

- **Module**
  - binarySwitch
  - audioVolume
  - televisionChannel
  - audioVideoInput
  - mediaSourceList

- **Sub-device**
  - ...

---

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<Domain id="org.onem2m.home"
xmlns="http://homegatewayinitiative.org/xml/dal/3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Modules>
    <ModuleClass name="binarySwitch"/>
    <ModuleClass name="audioVolume"/>
  </Modules>
  <Devices>
    <Device id="television">
      <Properties>
        <Property name="aProperty">
          <SimpleType type="string"/>
        </Property>
      </Properties>
      <Modules>
        <Module name="binarySwitch" extends class="binarySwitch" domain="org.onem2m.home"/>
        <Module name="audioVolume" extends class="audioVolume" domain="org.onem2m.home"/>
      </Modules>
    </Device>
  </Devices>
</Domain>
```

By: Dongjoo Kim, LG
THANK YOU!

a.kraft@telekom.de