RFP 185 - Abstraction Layer with oneM2M Smart Device Template

Draft

11 Pages

Abstract

This document describes the need of using oneM2M device abstraction layer [1]. in OSGi platform, a comparison with the current OSGi Device Abstraction Layer [2]., and the major requirements for the integration of oneM2M standard concepts.

0 Document Information

0.1 Table of Contents

0.2 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in 6.1.

`Source code is shown in this typeface.`

0.3 Revision History

The last named individual in this history is currently responsible for this document.

Revision	Date	Comments
Initial	Sep14, 2016	Initial version Maciej Goluch, Orange Polska S.A, maciej.goluch@orange.com

Revision	Date	Comments
0.01	Sept. 20, 2016	Refinement of the content. Andre Bottaro, Orange
0,02	Oct. 10, 2016	Improvement of the comparison table, use cases, requiremnets. Submission to OSGi git repository with number 185 Maciej Goluch, Andre Bottaro, Orange

1 Introduction

OSGi technology is more and more popular in residential and Internet of Things solutions. Many devices use various protocols on the market which lead to the development of complex smart systems. In order to provide a convenient programming model, to make the smart solutions integration with these devices easier, the OSGi Alliance has specified and published a chapter named Device Abstraction Layer within OSGi Residential Specification Release 6 [2]..

However, OSGi Device Abstraction Layer adds a new abstraction model that is not backed by any software community or standard organization. The main concepts of OSGi current Device Abstraction layer are close to the ones of the Smart Device Template made by the Home Gateway Initiative and now adopted by oneM2M organization [1].. Adopting the Smart Device Template (SDT) is an opportunity to reach the audience of a strong organization, whose specifications are being implemented by open source communities.

This RFP describes the need to back OSGi device abstraction layer with oneM2M SDT standard, compares both models and defines the requirements to integrate this solution with OSGi Alliance.

2 Application Domain

Currently, IoT devices use various communication protocols. To provide a good support for third party developers, many organizations and companies create their own abstraction model of devices. So many abstraction layers exist in IoT application domain.

The application domain is well described in RFP 147 [3].. The picture from the RFP 147 on Fig 1 shows a reference architecture which is valid in the majority of cases under consideration.

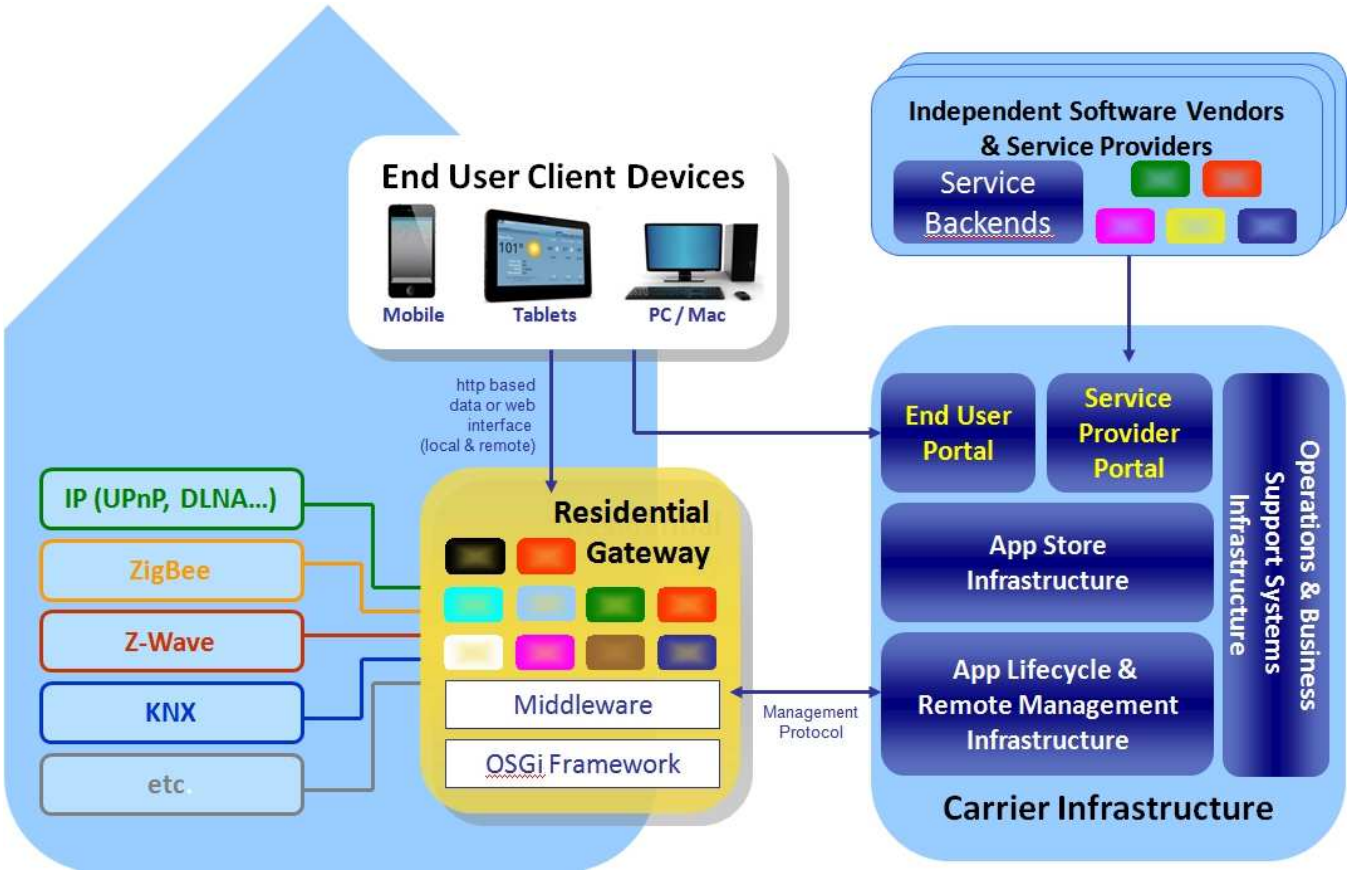

Fig 1: Abstraction layer in the Smart Home environment

The existing OSGi specifications which address related topics are:

Device Access Specification – focuses on the dynamic discovery of the proper driver when a new device is attached/connected to the residential gateway. The specification is limited to the driver installation needs.

UPnP™ Device Service Specification – defines an OSGi API to work with UPnP devices accessible on the home network

EnOcean Device Service Specification – defines an OSGi API to work with EnOcean devices accessible on the home network.

Device Abstraction Layer – unifies the work with devices provided by different protocols.

Device Abstraction Layer Functions Specification – defines concrete device function interfaces

2.1 Terminology + Abbreviations

Item	Description
Device Abstraction Layer	Unifies the work with devices provided by different protocols.
Smart Device Template	The SDT (SmarT Device Template) is an initiative from HGI to find consensus amongst various SDOs and industry alliances to derive a common approach for device modelling.

3 Problem Description

Many organizations and companies have created their own abstraction model of devices. Disunited solutions without common information model brings to a lack of interoperability between these systems. Today, third party application developers have to implement several abstraction layers to allow their application to be platform independent. They especially have to implement oneM2M Smart Device Template and information model which are now emerging as reference technologies in the IoT domain.

In order to avoid this situation and provide more convenient and standard programming model, the OSGi Alliance should adopt a standard device model, which is agreed by almost all IoT and M2M players. It could be achieved by using the standard abstraction model defined by oneM2M, Smart Device Template (SDT). SDT is publicly accessible [1]. An additional document (TR-0017 [4].) helps in the use of SDT.

One aim of the OSGi Device Abstraction layer (DAL) was initially to converge with the information model of the Home Gateway initiative (see OSGi RFP 147) that has been later adopted by oneM2M. The Home Gateway initiative has however specified the Smart Device Template after the OSGi Alliance has finalized its own layer. DAL and SDT specifications are now distinct. They fortunately share many high-level concepts while some of them and many details differ. The table below shows a comparison between the two models.

SDT	DAL
Domain • A unique name which acts like a namespace (e.g., "org.oneM2M.home.modules"). It is set by the organization creating the model. • Allowing reference to a package of definitions for the contained ModuleClasses and Device models.	**No equivalent concept.** The Java package naming is an opportunity to distinguish namespaces, which is not seized by the current OSGi DAL specification.
ModuleClass • Specifies a single service (e.g., audioVolume, binarySwitch) with one or more Actions, Properties, DataPoints and Events. • Every service that is described as a ModuleClass can be re-used in many Devices.	**Function** • atomic functional entity like switch or sensor. • The function belongs to a device. • The function provides a set of properties and operations
Module • Instantiation of a ModuleClass for a specific Device or SubDevice.	
Device • Physical, addressable, identifiable appliance, sensor and actuator • Contains one or more Modules, Properties and SubDevices	**Device** • Represents the device in the OSGi service registry • It is described with a set of service properties and functions.

SubDevice • SubDevice is a device which may be embedded in a Device and/or is addressed via another Device.	There is no equivalent concept in OSGi DAL. It might be mappable with the Device and keep relationship through service methods and service properties, like it is done for UPnPDevice services and their device children.
DataPoint • DataPoint shall be used to represent stateless operations • Functional information • E.g. targetTemperature in a thermostat, targetVolume in a television.	**FunctionData or Function Properties** • data structure which carries Function property value with extra metadata. • To be noted: an ambiguity between Functiondata and function property in OSGi DAL.
Action • shall be used when describing stateful condition, handling unknown internal state conditions, e.g., upVolume/downVolume by increasing/decreasing the audioVolume in steps, handling transactional procedures, or checking integrity using a user name and a password at the same time	**Function Operations** • Function operations are the main callable units. • They can be used by the applications to control the device • Synchronous or asynchronous operations can trigger different actions • As a result, a Function property can be updated
Arg • represents the parameter information which a device needs to carry out in a required "Action".	**Java methods arguments** Function Operations have zero or more arguments and zero or one return value.
Property • Device, SubDevice or ModuleClass could have Properties • Non-functional information • E.g. version, id and etc.	**Function service properties and Device service properties** • services can be filtered and accessed with their properties • Allows to define non-functional prop.: service name, version etc. and functional properties for Devices: status. • Can be used to represent different relationship between devices
	PropertyMetadata • contain metadata about the Function properties **OperationMetadata** • contain metadata about the Function operations
Event • elements are needed for automation protocols which "push" information, instead of relying on polling by the software application. • can be signalled ("published") by devices asynchronously	**FunctionEvent** • Asynchronous event. • It is posted through EventAdmin service and notifies for Function property change.
DataType • can be simple integers or string text, or rather complex • SimpleType • Boolean, Byte, Integer, Float, String, Enum, Date, Time, Datetime, Blob, Uri • Struct	**Types in DAL** • Java primitive type or corresponding reference type. • java.lang.String. • Numerical type i.e. the type which extends java.lang.Number. • Beans, but the beans properties must use

• Array	those rules. Java Beans are defined in JavaBeans specification. • java.util.Maps. The keys can be java.lang.String. The values of a single type follow these rules. • Arrays of defined types.
Potential possibility to represent the OSGi lifecycle using CRUD operations in oneM2M.	**Lifecycle** • There is a possibility to install,/uninstall, start/stop bundles. • The bundles could dynamically register/unregister services. • Function services are registered before the Device service
Device models • Definition of devices, e.g. deviceLight, deviceOven	There is no particular devices specification in DAL, only a specification of basic functions exist.
Doc • contain the human-readable information • Each elements could have a Doc description	Documentation of the elements is provided in the Chapter 142 of the main specification document

The Smart Device Template is depicted with a UML diagram in oneM2M specification documents, see Fig 3.

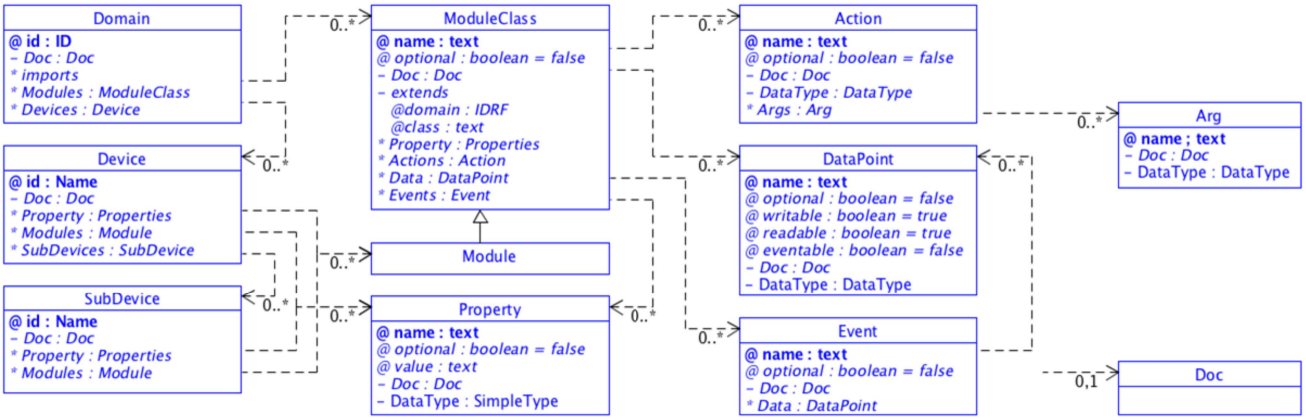

Fig 2: Smart Device Template diagram

To sum up the best idea is to provide a solution which is common, agreed by almost all of IoT players and provide many devices and reusable device functions models specifications. It could be all done by implementing the Smart Device Template and taking all advantages of it, including a set of Device and ModuleClass models specifications. The picture in Fig 3shows a general possible architecture of implementation which is based on the architecture proposed in RFP-147.

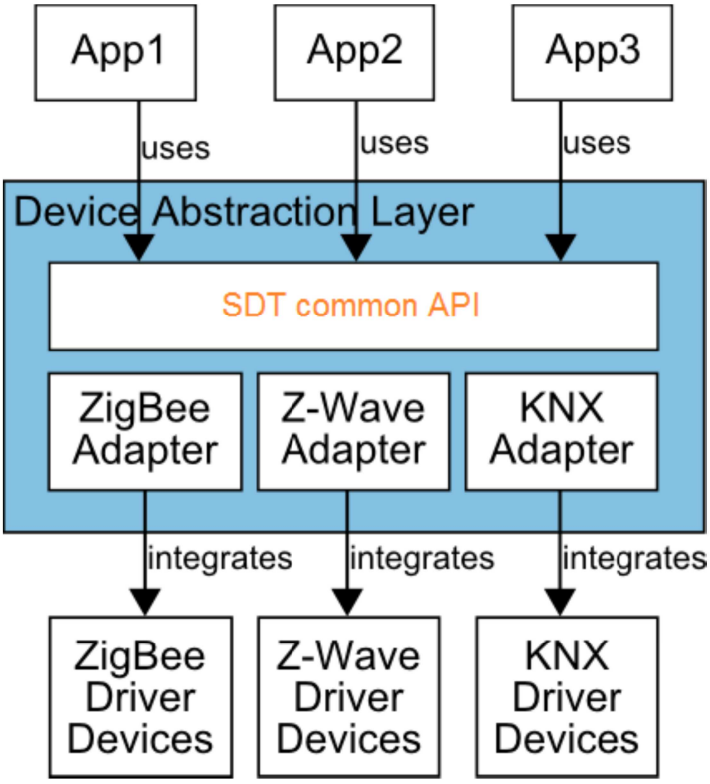

Fig 3: Tthe Smart Device Template abstraction layer in RFP 147 implementation schema.

4 Use Cases

The Device Abstraction Layer with Smart Device Template will first enable the same main use cases as the ones of the current Device Abstraction Layer. Here is a reminde r of RFP 147 main use cases that are still relevant:

Use case 1 – easy extension

A given line of residential gateways supports KNX and ZigBee and will run an application for home automation which controls all connected lights, heating elements and shutters. These residential gateways will be extended in the near future with support for EnOcean sensors/actuators. The home automation application is programmed in such a way that no software modification will be needed to support this new device protocol when it's available.

Use case 2 – common abstraction

A provider of a surveillance service is developing an application which must run on residential gateways from different vendors for controlling cameras also from different vendors not known in advance. This application is configurable so that it can use preinstalled local sensors (which exact type is not known in advance) in order to start streaming or recording.

Use case 3 – remote access

The residential gateway supports Z-Wave, KNX and EnOcean and will run an application to provide remote access to all connected devices. The application is designed in such way that no knowledge of device specific protocols is required for the remote access.

Use case 4 – dynamic network

The residential gateway supports Z-Wave and ZigBee and will run an application to track different device statistics. The gateway is installed in environment, where devices are often removed and added to the networks. The application is designed in such way that can easily track the new devices and release the resources linked to the removed devices.

Use case 5 – eventing

The residential gateway supports ZigBee and Z-Wave and will run an applications to monitor the temperature with the help of sensors. Usually, the temperature sensors are power saving devices and only reports periodically the temperature measurements. When the temperature limits are reached, air conditioners are activated. The application is designed in such way that can easily monitor the temperature measurements.

The adoption of Smart Device Template will facilitate the following use cases:

Use case 6 – easier integration

A provider would like to move their application to OSGi from another system. A common, well known device abstraction model can help developers in the implementation phase if both would use it. Device abstraction model which is agreed by a lot of M2M integrators and IoT players, will be implemented by other systems and databases providers, etc.

Use case 7 – improving product time to market

A provider would like to very quickly create a flexible, easy extensible system. A solution with many devices and functions models already defined is then helpful.

Use case 8 – smart home provider cooperation

Two families which lives on different countries use a smart home systems from various vendors. One of them would like to give the control of their smart home to another one cause going for vacation. It is easier to perform due to the fact that both vendors use a common device abstraction layer.

Use case 9 – definition of distinct device domains

An IoT gateway integrates devices from different vendors. Some devices are mappable on oneM2M home information model, some are described with UPnP Device Control Protocols (e.g, a UPnP Media Server) that has no direct mapping with available oneM2M devices and module classes, some are with vendor-specific information models. All devices are mapped on oneM2M Smart Device Template with devices and modules. The developers distinguish the various set of devices with distinct oneM2M domains.

5 Requirements

Requirement 1. The solution MUST enable the discovery and control of devices which is applicable for any relevant device protocol.

Requirement 2. The solution MUST define an API for the control of devices, which is independent from the device protocols.

Requirement 3. The solution MUST include device access control based on user and application permissions compliant with the OSGi security model.

Requirement 4. The solution MUST take advantage of the security features available in the device protocols.

Requirement 5. The solution MUST include a device protocol independent notification mechanism realized according to the OSGi event mechanisms.

Requirement 6. The solution MUST map all oneM2M SDT concepts.

Requirement 7. The solution MUST enable the implementation of oneM2M home information model.

Requirement 8. The solution MAY be a new version of the existing DAL solution aligned with SDT or it MAY be completely new solution based on SDT.

Requirement 9. The solution MUST register device or/and device related instances to the OSGi service registry.

Requirement 10. The solution MAY replace OSGi Device Abstraction Layer Functions Specification with a Java OSGi representation of devices and module classes specified in oneM2M home information model [1]..

Requirement 11. The solution MUST represent oneM2M domain concept.

6 Document Support

6.1 References

[1]. OneM2M, Home_Appliances_Information_Model_and_Mapping, TS-0023-V2.0.0, August 2016

[2]. OSGi Alliance, OSGi Residential Specification Release 6, July 2015

[3]. OSGi Alliance, RFP 147 Device Abstraction Layer, December 2012

[4]. OneM2M, Home Domain Abstract Information Model, TR-0017-V2.0.0, August 2016.

6.2 Author's Address

Name	Andre Bottaro
Company	Orange
Address	28 Chemin du Vieux Chene, Grenoble, France
e-mail	andre.bottaro@orange.com

Name	Maciej Goluch
Company	Orange Polska S.A.
Address	Obrzezna 7, Warsaw, Poland
e-mail	Maciej.goluch@orange.com

6.3 End of Document