



OSGiTM
Alliance

RFP-175 Device Access Specification Update

Draft

6 Pages

Abstract

Some points where the Device Access Specification might be improved for usability

Copyright © OSGi Alliance 2016.

All company, brand and product names contained within this document may be trademarks that are the sole property of the respective owners.

The above notice must be included on all copies of this document that are made.

1 Document Information

1.1 Table of Contents

1 Document Information.....	2
1.1 Table of Contents.....	2
1.2 Terminology and Document Conventions.....	2
1.3 Revision History.....	2
2 Introduction.....	3
3 Application Domain.....	3
3.1 Terminology + Abbreviations.....	3
4 Problem Description.....	3
5 Use Cases.....	4
5.1 Base Driver implementation.....	4
6 Requirements.....	5
7 Document Support.....	5
7.1 References.....	5
7.2 Author's Address.....	6
7.3 End of Document.....	6

1.2 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 7.1.

Source code is shown in this typeface.

1.3 Revision History

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

Revision	Date	Comments
Initial	Apr 28 2015	Initial version.

Revision	Date	Comments
	Jan 12 2016	Removed implementation details, according to the feedbacks received in the Cologne OSGi F2F (2014).

2 Introduction

This Device Admin Enhancements RFP suggests a set of new ways for using the Device Admin Specification with the final purpose to reduce as much as possible the burden necessary to implement a Driver service.

3 Application Domain

The Device Access Specification defines a model for handling devices on a OSGi Service Platform. Although the specification is very complete, it is also pretty old and it was never modified since the OSGi R3 Service Platform where first has been released. This RFP tries to highlight some aspects where the Device Access Specification 1.0 could be improved by taking into account of some concepts that were not available in OSGi at that time.

3.1 Terminology + Abbreviations

DS	Declarative Services;
DAS	Device Access Specification
POJO	Plain Old Java Object; this term is used in OSGi for representing java classes that do not include any reference to OSGi core framework classes or interfaces.
SCR	Service Components Runtime; this terminology is used to represent a Declarative Service implementation.

4 Problem Description

The Device Access specification defines two entities, the *device* and the *driver*. A *device attachment algorithm* is formalized and used by a Device Manager entity uses to find out more suitable driver for a specific device. In one is found it is then *attached* to the device. In the DAS *devices* and *drivers* are represented by services registered in the framework:

- A Device Service (see 6 paragraph 103.2) can be either be a service of interface `org.osgi.service.Device` or any kind of service with a “*DEVICE_CATEGORY*” service property defined.
- A Driver Service MUST register a service of type `org.osgi.service.Driver` and a `DRIVER_ID` service property.

As a consequence of the above statements, a *driver*, by design, cannot be a POJO because it has to register a service of interface `org.osgi.service.Driver`. Moreover this interface defines two methods `attach()` and `match()` that have an argument of type `org.osgi.framework.ServiceReference`.

On the other hand a *device* could be implemented as a POJO. The `org.osgi.service.Device` interface (if the device chose to implement it) has just a `noDriverFound()` method with no parameters. If the device implementer doesn't need to be notified about the “no driver found” event, by the device manager, a device could be implemented as described before without any dependence with the DAS API.

Another drawback of the current Device Access Specification is that the Device Manager delegates to the attached driver service the burden to track the attached device service, using, for instance, a `ServiceTracker` (see section 103.4.4 'Driver Service Unregistration' of 6).

The last, but not least limitation of DAS version 1.1, is that a Driver Service object, must be potentially be instantiated, even if it is eventually this driver not actually chosen by the Device Manager attachment algorithm.

This RFP has the purpose of providing some a use case that show how the developer may take advantage of a modification of the way the Device Access specification may be used by an OSGi developer.

This RFP and therefore the described use cases, doesn't involve the *Driver Locator Service* (see 6 paragraph 103.5) and the *Driver Selector Service* (see 6 paragraph 103.6).

5 Use Cases

5.1 Base Driver implementation

A home automation OSGi application is installed on a gateway. This application is composed by different base drivers, one for each of the Home Automation technologies supported by the software: ZigBee, Zwave, EnOcean. The end user may enable the gateway to handle physical devices of one of these technologies, by plugging in the USB port the relative external dongle.

The base driver that is able to use this dongle, is automatically attached by the Device Manager. If the dongle is not supported by any of the installed base drivers, nothing happens.

We suppose to use an implementation of the Serial Device Service specification (see chapter 147 of 6), since we suppose that all these dongles are seen by the operating system as USB serial port devices. We suppose also that this implementation register also the service with the USBInfoDevice interface (see the USB Information Device Category Specification, chapter 146 of 6).

When a dongle is plugged in the USB port, this Serial Device implementation register a SerialDevice service, at least with the following properties:

Service property	Value	Description
DEVICE_CATEGORY	[Serial, USBInfo]	The Device is a Serial device, and an USB device at the same time.
serial.comport	e.g. /dev/ttyUSB0 or COM1	The underlying Operating System com device port.
USB_IDVENDOR	e.g. 0202	The USB vendor ID of the dongle
USB_IDPRODUCT	e.g. AFBD	The USB device ID of the dongle

When the Device Manager detects this service, it recognizes it as a Device service and starts to find-out a suitable Driver from those registered in the OSGi framework.

Both the ZigBee, Zwave and EnOcean base drivers are Driver Services, so that they needs to be instantiated so that the Device Manager may call the Driver.match() method on them. At the end of the process at most one of them is chosen and attached to the SerialDevice device.

The attached base driver should then take care of tracking the SerialDevice service, because it have to release this service when it disappears. This may happen when the bundle that register this service is stopped or when the USB dongle is unplugged by the user or when the driver service itself is unregistered (for instance because the bundle it belongs is stopped).

6 Requirements

- R-1 The solution **MUST** be backward compatible. Any OSGi bundle compliant with the Device Access Specification Version 1.1 must continue to work.
- R-2 The solution **MUST** allow to implement a driver service as a POJO.
- R-3 The solution **SHALL** enforce the service lazy activation paradigm by providing a means to avoid the Driver service object instantiation with the purpose of only issuing the match() method.
- R-4 The solution **MUST** define a solution to avoid the developer to explicitly track attached device services with the purpose to be notified when these services are unregistered (attach e detach methods).
- R-5 The solution **MAY** allow to change the names of the methods used for attaching and detaching a device service.

Draft

January 12, 2016

- R-6 The solution MAY require a change in the Device Access Specification, including adding new interface definitions in addition to the currently defined in the 1.1.
- R-7 The solution MAY require a change in the Declarative Services Specification.

7 Document Support

7.1 References

- [1] Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, RFC2119, March 1997.
- [2] Software Requirements & Specifications. Michael Jackson. ISBN 0-201-87712-0
- [3] OSGi Service Platform Service Compendium Release 4, Version 4.3 Device Access Specification, Version 1.1
- [4] OSGi Residential Release 6, July 2015.

7.2 Author's Address

Name	Nicola Portinaro
Company	Telecom Italia
Address	Via Guglielmo Reiss Romoli 274, 10148 Torino, Italy
Voice	+39 331 600 1153
e-mail	nicola.portinaro@telecomitalia.it

7.3 End of Document