Device Access Specification Update

Draft

6 Pages

Abstract

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

1 Document Information

1.1 Table of Contents

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. Nicola Portinaro, Telecom Italia, nicola.portinaro@telecomitalia.it

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 integrate a Driver service in this architecture.

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 Acces 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 5 paragraph 103.2) can be either be a service of interface `org.osgi.service.Device` or any kind of service but with a "*DEVICE_CATEGORY"* service property
- 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 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 5)..

The last, but not least limitation of DAS version 1.1, is that a Driver Service object, must 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 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 cover the Driver Locator Service concept (see 5 paragraph 103.5) and the Driver Selector Service concept (see 5 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 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.

In order to implement the scenario above described, the Serial Device Service specification is used, since all this dongles implement a USB serial protocol. This specification registers in the framework a service of interface SerialDevice with the following properties:

Property	Value	Description
DEVICE_CATEGORY	[Serial, USB]	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 comport.
VID	0x0202	The USB vendor ID of the dongle
DID	0xAFBD	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. 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 match() method on them. Eventually, only one of them will be chosen and attached to the SerialDevice service. This base driver should be 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 dongle is unplugged by the user.

6 Requirements

R-1 The solution MUST be backward compatible. Any bundle developed to work with the Device Access Specification Version 1.1 or DS Version <= 1.6 should continue to work.

R-2 The solution MUST provide a mechanism to implement a Driver Service in a way similar to what SCR does.

R-3 The solution MUST define a mechanism to automatically track attached devices and notify an attached driver about the device unregistrations.

R-4 The solution MUST use java reflection to find the methods called by the Device Manager during the Device match, attachment and detachment process.

R-5 The solution MUST permit to change the default names for the methods that a Driver implements.

R-6 The solution MAY require a change of the Device Access Specification, including adding new interface flavors for the Driver and Device services.

R-7 The solution MAY require a change of the Declarative Services Specification

R-8 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 calling the match() method.

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

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