RFC 202 - USB Information Device Category

Final

30 Pages

Abstract

This document defines the device category for USB devices information in OSGi.

0 Document Information

0.1 License

DISTRIBUTION AND FEEDBACK LICENSE, Version 2.0

The OSGi Alliance hereby grants you a limited copyright license to copy and display this document (the “Distribution”) in any medium without fee or royalty. This Distribution license is exclusively for the purpose of reviewing and providing feedback to the OSGi Alliance. You agree not to modify the Distribution in any way and further agree to not participate in any way in the making of derivative works thereof, other than as a necessary result of reviewing and providing feedback to the Distribution. You also agree to cause this notice, along with the accompanying consent, to be included on all copies (or portions thereof) of the Distribution. The OSGi Alliance also grants you a perpetual, non-exclusive, worldwide, fully paid-up, royalty free, limited license (without the right to sublicense) under any applicable copyrights, to create and/or distribute an implementation of the Distribution that: (i) fully implements the Distribution including all its required interfaces and functionality; (ii) does not modify, subset, superset or otherwise extend the OSGi Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the OSGi Name Space other than those required and authorized by the Distribution. An implementation that does not satisfy limitations (i)-(ii) is not considered an implementation of the Distribution, does not receive the benefits of this license, and must not be described as an implementation of the Distribution. "OSGi Name Space" shall mean the public class or interface declarations whose names begin with "org.osgi" or any recognized successors or replacements thereof. The OSGi Alliance expressly reserves all rights not granted pursuant to these limited copyright licenses including termination of the license at will at any time.

EXCEPT FOR THE LIMITED COPYRIGHT LICENSES GRANTED ABOVE, THE OSGi ALLIANCE DOES NOT GRANT, EITHER EXPRESSLY OR IMPLIEDLY, A LICENSE TO ANY INTELLECTUAL PROPERTY IT, OR ANY THIRD PARTIES, OWN OR CONTROL. Title to the copyright in the Distribution will at all times remain with the OSGi Alliance. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted therein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

THE DISTRIBUTION IS PROVIDED "AS IS," AND THE OSGi ALLIANCE (INCLUDING ANY THIRD PARTIES THAT HAVE CONTRIBUTED TO THE DISTRIBUTION) MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DISTRIBUTION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.
NEITHER THE OSGi ALLIANCE NOR ANY THIRD PARTY WILL BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE DISTRIBUTION.

Implementation of certain elements of this Distribution may be subject to third party intellectual property rights, including without limitation, patent rights (such a third party may or may not be a member of the OSGi Alliance). The OSGi Alliance is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

The Distribution is a draft. As a result, the final product may change substantially by the time of final publication, and you are cautioned against relying on the content of this Distribution. You are encouraged to update any implementation of the Distribution if and when such Distribution becomes a final specification.

The OSGi Alliance is willing to receive input, suggestions and other feedback (“Feedback”) on the Distribution. By providing such Feedback to the OSGi Alliance, you grant to the OSGi Alliance and all its Members a non-exclusive, non-transferable,

worldwide, perpetual, irrevocable, royalty-free copyright license to copy, publish, license, modify, sublicense or otherwise distribute and exploit your Feedback for any purpose. Likewise, if incorporation of your Feedback would cause an implementation of the Distribution, including as it may be modified, amended, or published at any point in the future ("Future Specification"), to necessarily infringe a patent or patent application that you own or control, you hereby commit to grant to all implementers of such Distribution or Future Specification an irrevocable, worldwide, sublicenseable, royalty free license under such patent or patent application to make, have made, use, sell, offer for sale, import and export products or services that implement such Distribution or Future Specification. You warrant that (a) to the best of your knowledge you have the right to provide this Feedback, and if you are providing Feedback on behalf of a company, you have the rights to provide Feedback on behalf of your company; (b) the Feedback is not confidential to you and does not violate the copyright or trade secret interests of another; and (c) to the best of your knowledge, use of the Feedback would not cause an implementation of the Distribution or a Future Specification to necessarily infringe any third-party patent or patent application known to you. You also acknowledge that the OSGi Alliance is not required to incorporate your Feedback into any version of the Distribution or a Future Specification.

I HEREBY ACKNOWLEDGE AND AGREE TO THE TERMS AND CONDITIONS DELINEATED ABOVE.

0.2 Trademarks

OSGi™ is a trademark, registered trademark, or service mark of the OSGi Alliance in the US and other countries. Java is a trademark, registered trademark, or service mark of Oracle Corporation in the US and other countries. All other trademarks, registered trademarks, or service marks used in this document are the property of their respective owners and are hereby recognized.

0.3 Feedback

This document can be downloaded from the OSGi Alliance design repository at https://github.com/osgi/design
The public can provide feedback about this document by opening a bug at https://www.osgi.org/bugzilla/.

0.4 Table of Contents

0.5 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 1.

`Source code is shown in this typeface.`

0.6 Revision History

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

Revision	Date	Comments
Initial	April 10, 2013	Initial version Yukio Koike, NTT Corporation, koike.yukio@lab.ntt.co.jp
v0.2	July 4, 2013	– added RFC number to title – added 5.1.1.1 Optional Device Access Category – modified 5.2.2 Service properties from USB Specification Yukio Koike, NTT Corporation, koike.yukio@lab.ntt.co.jp

Revision	Date	Comments
v0.3	Sept. 9, 2013	– modified based on the F2F meeting in Paris Yukio Koike, NTT Corporation, koike.yukio@lab.ntt.co.jp
v0.4	Nov. 19, 2013	– modified based on the F2F meeting in Hursley Yukio Koike, NTT Corporation, koike.yukio@lab.ntt.co.jp
v0.5	Nov. 19, 2013	– Updated Javadoc section Yukio Koike, NTT Corporation, koike.yukio@lab.ntt.co.jp
v0.6	Feb. 17, 2014	– Modified based on the F2F meeting in Sofia – restructured chapters – added Preconditions and Behavior Yukio Koike, NTT Corporation, koike.yukio@lab.ntt.co.jp
v0.7	April 7, 2014	Modified based on the F2F meeting in Cologne. – Added operation example details and added USB serial example 2, USB storage example 3 – Added Considered Alternatives (8.1) – Modified some wording – Modified some service properties' Java type Yukio Koike, NTT Corporation, koike.yukio@lab.ntt.co.jp
v0.8	May 1, 2014	– Modified about USBDevice service registration rule – Changed treatment about bInterfaceClass/ SubClass/ Protocol – Changed some service properties' Java types – Added some service properties – Modified Security Considerations – English corrections Yukio Koike, NTT Corporation, koike.yukio@lab.ntt.co.jp
v0.9	Sept. 18, 2014	– Changed RFC title and category/interface/package names based on F2F meeting in Madrid – Changed match scale and some property key based on F2F meeting in Basel – Changed RFC templete to a new one – Replaced Fig. 3 (Class Diagram) – Add a match scale (MATCH_VERSION) Yukio Koike, NTT Corporation, koike.yukio@lab.ntt.co.jp
v0.10	Nov. 6, 2014	– Removed Serial / Mass Storage sections Yukio Koike, NTT Corporation, koike.yukio@lab.ntt.co.jp
final	Dec. 15, 2014	– Finalized to vote Yukio Koike, NTT Corporation, koike.yukio@lab.ntt.co.jp

1 Introduction

OSGi Device Access Specification defines a unified and sophisticated way to handle devices attached to a residential gateway or devices found in the home network by using various protocols such as USB, Zigbee, ZWave, KNX, UPnP etc. However, OSGi Device Access Specification clearly declare that Device Category must be defined outside of OSGi Device Access Specification.

Recently, OSGi is gaining popularity as enabling technology for building embedded system in residential market. It gets popular that a HGW has USB interfaces and the needs of handling USB devices attached to a residential gateway is increased.

This RFC defines a device category for USB devices.

2 Application Domain

Currently there are several standardization bodies such as OSGiA, HGI, BBF, which deal with the deployment of services in an infrastructure based on the usage of a Residential Gateway running OSGi as Execution Platform.

In order to realize services which access not only IP devices but also non-IP devices connected to the residential gateway, there are several protocols for home networks, such as ZigBee, Z-Wave, KNX/EHS, ECHONET, ECHONET-LITE, etc.. While some residential gateways support those protocols on themselves, others do not. Many residential gateways have USB interfaces and there exist USB dongles which support those protocols. Therefore, there is a need to support those protocols using USB dongles attached to a residential gateway (Fig. 1). In addition, most of USB dongles can be controlled through Serial Communication.

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 it attached/connected to the residential gateway

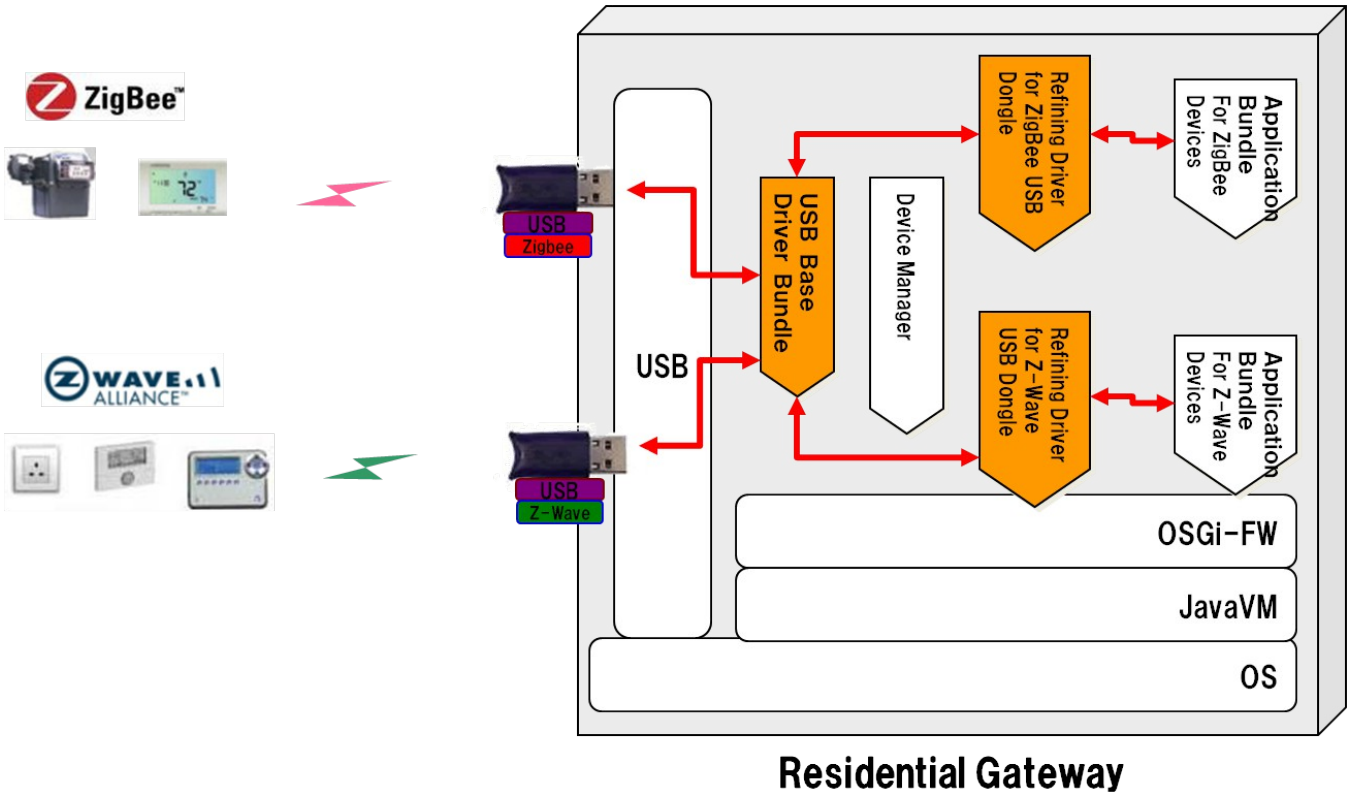

Fig 1 USB Dongles and Residential gateway

2.1 Terminology + Abbreviations

- Base Drivers: see "103.4.2.1" in OSGi Device Access Specification [3].
- Refining Drivers: see "103.4.2.2" in OSGi Device Access Specification [3].
- Match value: the value match() method of a Driver service registered by the refining driver bundle returns. Matching is explained in "103.7.2 The Device Attachment Algorithm" in OSGi Device Access Specification [3].
- Device Descriptor: see "9.6.1" in Universal Serial Bus Specification[4].

3 Problem Description

The existing OSGi Device Access Specification provides the unified way to installation and activation of driver bundles. However, the OSGi Device Access Specification declares the device category for specific devices must be defined outside of itself. Currently, no device category for USB devices has been defined yet.

The lack of the device category for USB devices causes the following problems.

[Problem 1] The developer of a refining driver bundle, which registers a Driver service at its activation, cannot design and implement Driver#attach(ServiceReference) method without knowledge of service properties set to the Device service registered by a USB base driver.

[Problem 2] The developer of a refining driver bundle, which registers a Driver service at its activation, cannot design and implement Driver#match(ServiceReference) method without knowledge of service properties set to the Device service registered by a USB base driver and without the definition of match values to be returned.

In other words, without the device category for USB devices, a refining driver bundle developed by developer A can cooperate with the USB base driver bundle developed by the same developer A but cannot cooperate with the USB base driver bundles developed by the different developer B.

4 Requirements

[REQ_1] The solution MUST be compatible with OSGi Device Access Specification .

[REQ_2] The solution MUST define the details of the registration of a Device service by a USB base driver bundle when a USB device is attached.

[REQ_2-1] The solution MUST define the service interface under which the Device service is registered.

[REQ_2-2] The solution MUST define the service properties with which the Device service is registered: A set of service properties, their data types, and semantics, each of which must be declared as either MANDATORY or OPTIONAL.

[REQ_3] The solution MUST define the way how a driver bundle controls an attached USB device which can be controlled through Serial communication.

[REQ_4] The solution MAY define a range of match values specific to this device category.

[REQ_5] The range of match values MUST be sufficient to describe the required range of native serial drivers specified by the HGI, especially the following ones:

– Class drivers for Human Interface Device (HID) and Communications Device Class (CDC) [1]

– Drivers for FTDI Virtual Com Ports with a variable list of supported USB Vendor Identifiers and Product Identifiers[2].

– Drivers for Silicon Labs CP210x USB to UART bridge and CP2110 HID USB to UART bridge[3].

– USB drivers for Prolific PL-2303 USB to Serial Bridge Controller[4].

5 Technical Solution

USB information device category defines the following elements:

1. An interface that all devices belonging to this category must implement.
2. A set of service registration properties, their data types, and semantics, each of which must be declared as either MANDATORY or OPTIONAL for this device category.
3. A range of match values specific to this device category.

RFC 213 “Serial Device Service” defines the OSGi service for Serial devices. Therefore RFC 213 and this RFC are the solution for USB-Serial devices.

5.1 USBInfoDevice Service

The device services are registered in the OSGi service registry with `org.osgi.service.usbinfo.USBInfoDevice` interface. The service is registered by a USB information base driver bundle when a USB device is attached. A USB information base driver bundle must implement `org.osgi.service.USBInfoDeviceusbinfo.USBInfoDevice` interface and register the OSGi service under `org.osgi.service.usbinfo.USBInfoDevice`. Refining drivers can find USB devices via USBInfoDevice services and identify the device. The USBInfoDevice service has a set of properties.

Universal Serial Bus Specification (USB Specification) [4]. defines that a USB device has USB interface(s). A USB information base driver bundle must register USBInfoDevice services number of USB interfaces. A USBInfoDevice service has information that contains a USB device information and a USB interface information.

1 http://www.usb.org/developers/devclass_docs#approved for details of USB device classes

2 http://www.ftdichip.com/Drivers/VCP.htm

3 http://www.silabs.com/products/mcu/pages/usbtouartbridgevcpdrivers.aspx.

4 http://www.prolific.com.tw

Figure 2 shows the class diagram.

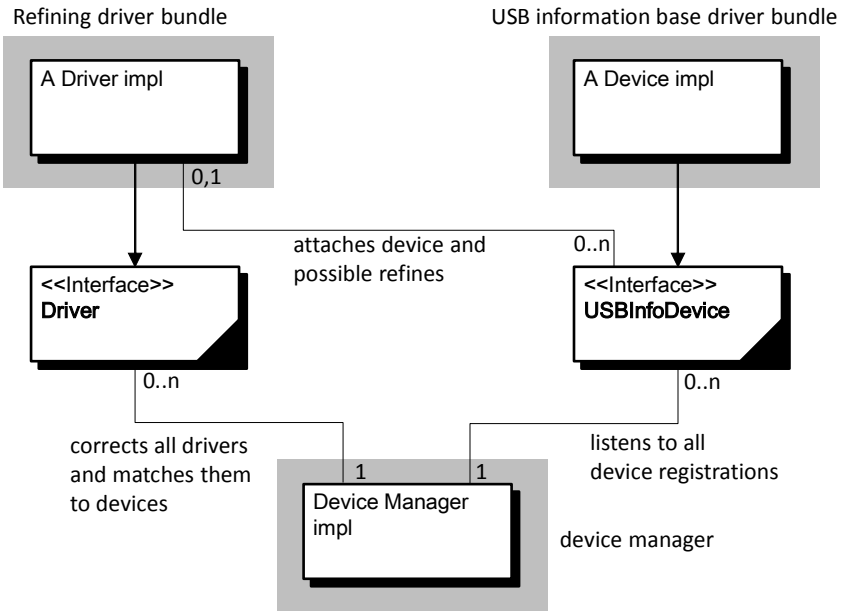

Fig 2:Class Diagram

5.1.1 Assumptions

The USB information base driver may need native drivers such as kernel drivers on Linux (Fig 3). This document has a precondition that there are native drivers. It is out of scope how to install native drivers.

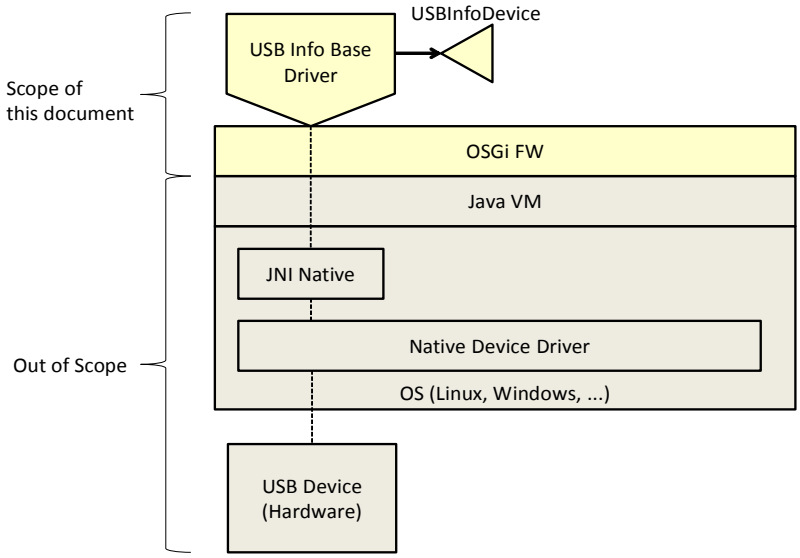

Fig 3:Software Structure and Scope

5.1.2 Device Access Category

The device access category is called "USBInfo". The category name is defined as a value of USBInfoDevice. `DEVICE_CATEGORY` constant. It can be used as a part of `org.osgi.service.device.Constants.DEVICE_CATEGORY` service key value. The category imposes the following specification rules.

- `USBInfoDevice.DEVICE_CATEGORY` - MANDATORY property. The value is "USBInfo". Constant for the value of the service property `DEVICE_CATEGORY` used for all USB devices. A USB information base driver bundle must set this property key.

5.1.3 Service properties from USB Specification

The USB Specification defines a device descriptor. USB devices report their attributes using descriptors. USBInfoDevice service has some properties from the USB device descriptor. Table 1 shows them.

Table 1: Device Descriptor and Service Property

Device Descriptor's Field from USB Spec.	**USBInfoDevice's service property**	**M/O**	**Java type**
bLength	`none`	-	-
bDescriptorType	`none`	-	-
bcdUSB	`usbinfo.bcdUSB`	O	`String`
bDeviceClass	`usbinfo.bDeviceClass`	M	`String`
bDeviceSubClass	`usbinfo.bDeviceSubClass`	M	`String`

bDeviceProtocol	`usbinfo.bDeviceProtocol`	M	`String`
bMaxPacketSize0	`usbinfo.bMaxPacketSize0`	O	`Integer`
idVendor	`usbinfo.idVendor`	M	`String`
idProduct	`usbinfo.idProduct`	M	`String`
bcdDevice	`usbinfo.bcdDevice`	M	`String`
iManufacturer	`usbinfo.Manufacturer`	O	`String`
iProduct	`usbinfo.Product`	O	`String`
iSerialNumber	`usbinfo.SerialNumber`	O	`String`
bNumConfigurations	`usbinfo.bNumConfigurations`	O	`Integer`

- `usbinfo.bcdUSB` - OPTIONAL property key. The value is `String`, the 4-digit BCD format.
  - Example: ”0210”
- `usbinfo.bDeviceClass` - MANDATORY property key. The value is `String`, hexadecimal, 2-digits.
  - Example: “ff”
- `usbinfo.bDeviceSubClass` - MANDATORY property key. The value is `String`, hexadecimal, 2-digits.
  - Example: ”ff”
- `usbinfo.bDeviceProtocol` - MANDATORY property key. The value is `String`, hexadecimal, 2-digits.
  - Example: ”ff”
- `usbinfo.bMaxPacketSize0` – OPTIONAL property key. The value is Integer.
- `usbinfo.idVendor` - MANDATORY property key. The value is `String`, hexadecimal, 4-digits.
  - Example: “0403”
- `usbinfo.idProduct` - MANDATORY property key. The value is `String`, hexadecimal, 4-digits.
  - Example: “8372”
- `usbinfo.bcdDevice` - MANDATORY property key. The value is `String`, the 4-digit BCD format.

- Example: ”0200”

- `usbinfo.Manufacturer` - OPTIONAL property key. The value is `String` of indicated in iManufacturer. (The value is not the index.)
  - Example: "Buffalo Inc."

- `usbinfo.Product` - OPTIONAL property key. The value is `String` of indicated in iProduct. (The value is not the index.)
  - Example: "USB2.0 PC Camera"

- `usbinfo.SerialNumber` - OPTIONAL property key. The value is `String` of indicated in iSerialNumber. (The value is not the index.)
  - Example: "57B0002600000001"

- `usbinfo.bNumConfigurations` – OPTIONAL property key. The value is Integer.

According to the USB Specification, a device descriptor has some interface descriptors.

Refining drivers need each interface descriptors' *bInterfaceClass, bInterfaceSubClass* and *bInterfaceProtocol* to identify devices. So these fields add to the service properties (see Table 2).

Table 2: Interface Descriptor and Service Property

Interface Descriptor's Field from USB Spec.	**USBInfoDevice's service property**	**M/O**	**Java type**
bLength	`none`	-	-
bDescriptorType	`none`	-	-
bInterfaceNumber	`usbinfo.bInterfaceNumber`	M	`Integer`
bAlternateSetting	`usbinfo.bAlternateSetting`	O	`Integer`
bNumEndpoints	`usbinfo.bNumEndpoints`	O	`Integer`
bInterfaceClass	`usbinfo.bInterfaceClass`	M	`String`
bInterfaceSubClass	`usbinfo.bInterfaceSubClass`	M	`String`
bInterfaceProtocol	`usbinfo.bInterfaceProtocol`	M	`String`
iInterface	`usbinfo.Interface`	O	`String`

- `usbinfo.bInterfaceNumber` – MANDATORY property key. The value is Integer.

- `usbinfo.bAlternateSetting` – OPTIONAL property key. The value is Integer.

- `usbinfo.bNumEndpoints` – OPTIONAL property key. The value is Integer.
- `usbinfo.bInterfaceClass` - MANDATORY property key. The value is `String`, hexadecimal, 2-digits.
  - Example: “ff”
- `usbinfo.bInterfaceSubClass` - MANDATORY property key. The value is `String`, hexadecimal, 2-digits.
  - Example: ”ff”
- `usbinfo.bInterfaceProtocol` - MANDATORY property key. The value is `String`, hexadecimal, 2-digits.
  - Example: ”ff”
- `usbinfo.Interface` - OPTIONAL property key. The value is `String` of indicated in iInterface. (The value is not the index.)

5.1.4 Other Service properties

Some other service properties are needed to identify and access a device by refining drivers.

Table 3: Other service properties

Service property	M/O	Java type
`usbinfo.bus`	M	`Integer`
`usbinfo.address`	M	`Integer`

- `usbinfo.bus` - MANDATORY property key. The value is `Integer`. Used to identify USB devices with same VID / PID. The value is the ID of the USB bus assigned when connecting the USB device. USB bus ID is integer. The USB bus ID does not change while the USB device remains connected.
  - Example: 3
- `usbinfo.address` - MANDATORY property key. The value is `Integer`. Used to identify USB devices with same VID / PID. The value is the ID of the USB address assigned when connecting the USB device. USB address is integer (001-127). The USB address does not change while the USB device remains connected.
  - Example: 2

5.1.5 Match scale

When the driver service is registered by the driver bundle, the Device Manager calls Driver#match() with the argument of the USBInfoDevice service's ServiceReference. The driver responds with the value based on below scale.

- `MATCH_VERSION` - Constant for the USB device match scale, indicating a match with `usbinfo.idVendor`, `usbinfo.idProduct` and `usbinfo.bcdDevice`. Value is 50.

- `MATCH_MODEL` - Constant for the USB device match scale, indicating a match with `usbinfo.idVendor` and `usbinfo.idProduct`. Value is 40.

- `MATCH_PROTOCOL` - Constant for the USB device match scale, indicating a match with `usbinfo.bDeviceClass`, `usbinfo.bDeviceSubClass` and `usbinfo.bDeviceProtocol`, or a match with usbinfo.bInterfaceClass, usbinfo.bInterfaceSubClass and usbinfo.bInterfaceProtocol. Value is 30.

- `MATCH_SUBCLASS` - Constant for the USB device match scale, indicating a match `usbinfo.bDeviceClass` and `usbinfo.bDeviceSubClass`, or a match with usbinfo.bInterfaceClass and usbinfo.bInterfaceSubClass. Value is 20.

- `MATCH_CLASS` - Constant for the USB device match scale, indicating a match with `usbinfo.bDeviceClass`, or a match with usbinfo.bInterfaceClass. Value is 10.

5.1.6 Operations

Figure 4 describes a mechanism to handle USB devices. When a USB device is attached, a USB information base driver bundle recognizes it via native device drivers and gets information from the USB device. The USB information base driver bundle registers a USBInfoDevice service with service properties from the information gained.

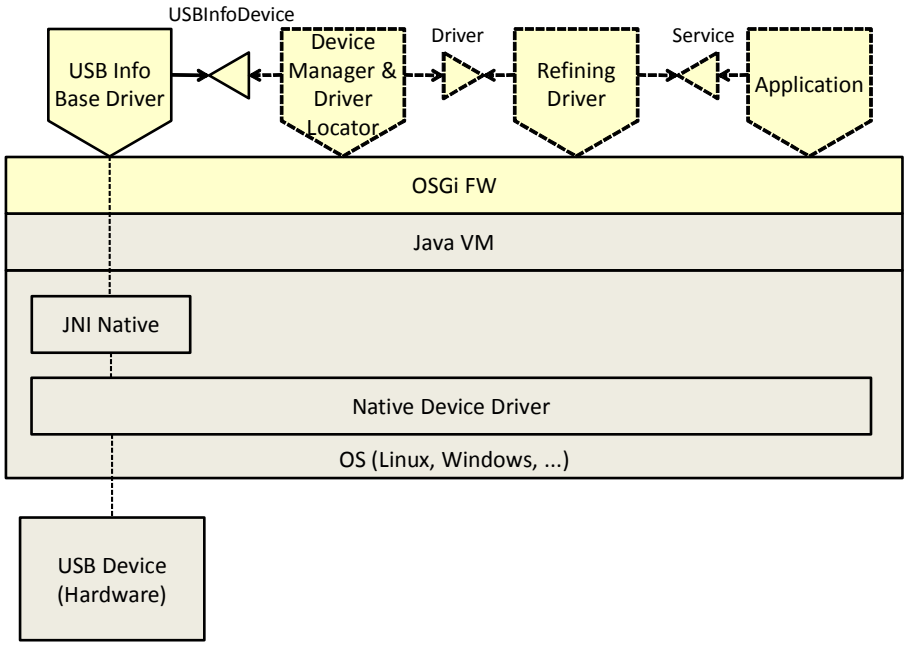

Fig 4:Device attachment example

6 Data Transfer Objects

This RFC will not provide Data Transfer Objects.

7 Javadoc

OSGi Javadoc

10/20/14 5:42 PM

Package Summary		Page
org.osgi.service.usbinfo	USB Information Device Category Specification Package Version 1.0.	18

Package org.osgi.service.usbinfo

USB Information Device Category Specification Package Version 1.0.

See:

Description

Interface Summary		Page
USBInfoDevice	Represents a USB device.	19

Package org.osgi.service.usbinfo Description

USB Information Device Category Specification Package Version 1.0.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

```
Import-Package: org.osgi.service.usbinfo; version="[1.0,2.0)"
```

Example import for providers implementing the API in this package:

```
Import-Package: org.osgi.service.usbinfo; version="[1.0,1.1)"
```

Interface USBInfoDevice

org.osgi.service.usbinfo

public interface **USBInfoDevice**

Represents a USB device. For each USB device, an object is registered with the framework under the USBInfoDevice interface. A USB information base driver must implement this interface. The values of the USB property names are defined by the USB Implementers Forum, Inc. The package name is org.osgi.service.usbinfo.

Field Summary		Page
String	**DEVICE_CATEGORY** MANDATORY property.	21
int	**MATCH_CLASS** Constant for the USB device match scale, indicating a match with usbinfo.bDeviceClass, or a match with bInterfaceClass in one of usbinfo.interfaceclasses.	26
int	**MATCH_MODEL** Constant for the USB device match scale, indicating a match with usbinfo.idVendor and usbinfo.idProduct.	25
int	**MATCH_PROTOCOL** Constant for the USB device match scale, indicating a match with usbinfo.bDeviceClass, usbinfo.bDeviceSubClass and usbinfo.bDeviceProtocol, or a match with bInterfaceClass, bInterfaceSubClass and bInterfaceProtocol in one of usbinfo.interfaceclasses.	25
int	**MATCH_SUBCLASS** Constant for the USB device match scale, indicating a matchusbinfo.bDeviceClass and usbinfo.bDeviceSubClass, or a match with bInterfaceClass and bInterfaceSubClass in one of usbinfo.interfaceclasses.	26
int	**MATCH_VERSION** Constant for the USB device match scale, indicating a match with usbinfo.idVendor, usbinfo.idProduct and usbinfo.bcdDevice.	25
String	**USB_ADDRESS** The key string of "usbinfo.address" service property. Used to identify USB devices with same VID / PID.	25
String	**USB_BALTERNATESETTING** Optional. The key string of "usbinfo.bAlternateSetting" service property. Service properties from USB Interface Descriptor.	23
String	**USB_BCDDEVICE** The key string of "usbinfo.bcdDevice" service property. Service properties from USB Device Descriptor.	22
String	**USB_BCDUSB** Optional. The key string of "usbinfo.bcdUSB" service property. Service properties from USB Device Descriptor.	21
String	**USB_BDEVICECLASS** The key string of "usbinfo.bDeviceClass" service property. Service properties from USB Device Descriptor.	21
String	**USB_BDEVICEPROTOCOL** The key string of "usbinfo.bDeviceProtocol" service property. Service properties from USB Device Descriptor.	21

String	**USB_BDEVICESUBCLASS** The key string of "usbinfo.bDeviceSubClass" service property. Service properties from USB Device Descriptor.	21
String	**USB_BINTERFACECLASS** The key string of "usbinfo.bInterfaceClass" service property. Service properties from USB Interface Descriptor.	24
String	**USB_BINTERFACENUMBER** The key string of "usbinfo.bInterfaceNumber" service property. Service properties from USB Interface Descriptor.	23
String	**USB_BINTERFACEPROTOCOL** The key string of "usbinfo.bInterfaceProtocol" service property. Service properties from USB Interface Descriptor.	24
String	**USB_BINTERFACESUBCLASS** The key string of "usbinfo.bInterfaceSubClass" service property. Service properties from USB Interface Descriptor.	24
String	**USB_BMAXPACKETSIZE0** Optional. The key string of "usbinfo.bMaxPacketSize0" service property. Service properties from USB Device Descriptor.	22
String	**USB_BNUMCONFIGURATIONS** Optional. The key string of "usbinfo.bNumConfigurations" service property. Service properties from USB Device Descriptor.	23
String	**USB_BNUMENDPOINTS** Optional. The key string of "usbinfo.bNumEndpoints" service property. Service properties from USB Interface Descriptor.	24
String	**USB_BUS** The key string of "usbinfo.bus" service property. Used to identify USB devices with same VID / PID.	25
String	**USB_IDPRODUCT** The key string of "usbinfo.idProduct" service property. Service properties from USB Device Descriptor.	22
String	**USB_IDVENDOR** The key string of "usbinfo.idVendor" service property. Service properties from USB Device Descriptor.	22
String	**USB_INTERFACE** Optional. The key string of "usbinfo.Interface" service property. Service properties from USB Interface Descriptor.	24
String	**USB_MANUFACTURER** Optional. The key string of "usbinfo.Manufacturer" service property. Service properties from USB Device Descriptor.	22
String	**USB_PRODUCT** Optional. The key string of "usbinfo.Product" service property. Service properties from USB Device Descriptor.	23
String	**USB_SERIALNUMBER** Optional. The key string of "usbinfo.SerialNumber" service property. Service properties from USB Device Descriptor.	23

Field Detail

DEVICE_CATEGORY

```
public static final String DEVICE_CATEGORY = "USBInfo"
```

MANDATORY property. The value is "USBInfo". Constant for the value of the service property DEVICE_CATEGORY used for all USB devices. A USB information base driver bundle must set this property key. See Also org.osgi.service.device.Constants.DEVICE_CATEGORY

USB_BCDUSB

```
public static final String USB_BCDUSB = "usbinfo.bcdUSB"
```

Optional.
The key string of "usbinfo.bcdUSB" service property. Service properties from USB Device Descriptor. Device Descriptor's Field from USB Spec is "bcdUSB". The value is String, the 4-digit BCD format. Example: "0210"

USB_BDEVICECLASS

```
public static final String USB_BDEVICECLASS = "usbinfo.bDeviceClass"
```

The key string of "usbinfo.bDeviceClass" service property. Service properties from USB Device Descriptor. Device Descriptor's Field from USB Spec is "bDeviceClass".
The value is String, hexadecimal, 2-digits. Example: "ff"

USB_BDEVICESUBCLASS

```
public static final String USB_BDEVICESUBCLASS = "usbinfo.bDeviceSubClass"
```

The key string of "usbinfo.bDeviceSubClass" service property. Service properties from USB Device Descriptor. Device Descriptor's Field from USB Spec is "bDeviceSubClass".
The value is String, hexadecimal, 2-digits. Example: "ff"

USB_BDEVICEPROTOCOL

```
public static final String USB_BDEVICEPROTOCOL = "usbinfo.bDeviceProtocol"
```

The key string of "usbinfo.bDeviceProtocol" service property. Service properties from USB Device Descriptor. Device Descriptor's Field from USB Spec is "bDeviceProtocol".
The value is String, hexadecimal, 2-digits. Example: "ff"

USB_BMAXPACKETSIZE0

```
public static final String USB_BMAXPACKETSIZE0 = "usbinfo.bMaxPacketSize0"
```

Optional.
The key string of "usbinfo.bMaxPacketSize0" service property.
Service properties from USB Device Descriptor. Device Descriptor's Field from USB Spec is "bMaxPacketSize0".
The value is Integer.

USB_IDVENDOR

```
public static final String USB_IDVENDOR = "usbinfo.idVendor"
```

The key string of "usbinfo.idVendor" service property.
Service properties from USB Device Descriptor. Device Descriptor's Field from USB Spec is "idVendor".
The value is String, hexadecimal, 4-digits.
Example: "0403"

USB_IDPRODUCT

```
public static final String USB_IDPRODUCT = "usbinfo.idProduct"
```

The key string of "usbinfo.idProduct" service property.
Service properties from USB Device Descriptor. Device Descriptor's Field from USB Spec is "idProduct".
The value is String, hexadecimal, 4-digits.
Example: "8372"

USB_BCDDEVICE

```
public static final String USB_BCDDEVICE = "usbinfo.bcdDevice"
```

The key string of "usbinfo.bcdDevice" service property.
Service properties from USB Device Descriptor. Device Descriptor's Field from USB Spec is "bcdDevice".
The value is String, the 4-digit BCD format.
Example: "0200"

USB_MANUFACTURER

```
public static final String USB_MANUFACTURER = "usbinfo.Manufacturer"
```

Optional.
The key string of "usbinfo.Manufacturer" service property.
Service properties from USB Device Descriptor. Device Descriptor's Field from USB Spec is "iManufacturer".
The value is String of indicated in iManufacturer. (The value is not the index.)
Example: "Buffalo Inc."

USB_PRODUCT

```
public static final String USB_PRODUCT = "usbinfo.Product"
```

Optional.
The key string of "usbinfo.Product" service property.
Service properties from USB Device Descriptor. Device Descriptor's Field from USB Spec is "iProduct".
The value is String of indicated in iProduct. (The value is not the index.)
Example: "USB2.0 PC Camera"

USB_SERIALNUMBER

```
public static final String USB_SERIALNUMBER = "usbinfo.SerialNumber"
```

Optional.
The key string of "usbinfo.SerialNumber" service property.
Service properties from USB Device Descriptor. Device Descriptor's Field from USB Spec is "iSerialNumber".
The value is String of indicated in iSerialNumber. (The value is not the index.)
Example: "57B0002600000001"

USB_BNUMCONFIGURATIONS

```
public static final String USB_BNUMCONFIGURATIONS = "usbinfo.bNumConfigurations"
```

Optional.
The key string of "usbinfo.bNumConfigurations" service property.
Service properties from USB Device Descriptor. Device Descriptor's Field from USB Spec is "bNumConfigurations".
The value is Integer.

USB_BINTERFACENUMBER

```
public static final String USB_BINTERFACENUMBER = "usbinfo.bInterfaceNumber"
```

The key string of "usbinfo.bInterfaceNumber" service property.
Service properties from USB Interface Descriptor. Interface Descriptor's Field from USB Spec is "bInterfaceNumber".
The value is Integer.

USB_BALTERNATESETTING

```
public static final String USB_BALTERNATESETTING = "usbinfo.bAlternateSetting"
```

Optional.
The key string of "usbinfo.bAlternateSetting" service property.
Service properties from USB Interface Descriptor. Interface Descriptor's Field from USB Spec is "bAlternateSetting".
The value is Integer.

USB_BNUMENDPOINTS

```
public static final String USB_BNUMENDPOINTS = "usbinfo.bNumEndpoints"
```

Optional.
The key string of "usbinfo.bNumEndpoints" service property.
Service properties from USB Interface Descriptor. Interface Descriptor's Field from USB Spec is "bNumEndpoints".
The value is Integer.

USB_BINTERFACECLASS

```
public static final String USB_BINTERFACECLASS = "usbinfo.bInterfaceClass"
```

The key string of "usbinfo.bInterfaceClass" service property.
Service properties from USB Interface Descriptor. Interface Descriptor's Field from USB Spec is "bInterfaceClass".
The value is String, hexadecimal, 2-digits.
Example: "ff"

USB_BINTERFACESUBCLASS

```
public static final String USB_BINTERFACESUBCLASS = "usbinfo.bInterfaceSubClass"
```

The key string of "usbinfo.bInterfaceSubClass" service property.
Service properties from USB Interface Descriptor. Interface Descriptor's Field from USB Spec is "bInterfaceSubClass".
The value is String, hexadecimal, 2-digits.
Example: "ff"

USB_BINTERFACEPROTOCOL

```
public static final String USB_BINTERFACEPROTOCOL = "usbinfo.bInterfaceProtocol"
```

The key string of "usbinfo.bInterfaceProtocol" service property.
Service properties from USB Interface Descriptor. Interface Descriptor's Field from USB Spec is "bInterfaceProtocol".
The value is String, hexadecimal, 2-digits.
Example: "ff"

USB_INTERFACE

```
public static final String USB_INTERFACE = "usbinfo.Interface"
```

Optional.
The key string of "usbinfo.Interface" service property.
Service properties from USB Interface Descriptor. Interface Descriptor's Field from USB Spec is "iInterface".

The value is String of indicated in iInterface. (The value is not the index.)

USB_BUS

```
public static final String USB_BUS = "usbinfo.bus"
```

The key string of "usbinfo.bus" service property.
Used to identify USB devices with same VID / PID. The value is the ID of the USB bus assigned when connecting the USB device. USB bus ID is integer. The USB bus ID does not change while the USB device remains connected.
The value is Integer.
Example: 3

USB_ADDRESS

```
public static final String USB_ADDRESS = "usbinfo.address"
```

The key string of "usbinfo.address" service property.
Used to identify USB devices with same VID / PID. The value is the ID of the USB address assigned when connecting the USB device. USB address is integer (001-127). The USB address does not change while the USB device remains connected.
The value is Integer.
Example: 2

MATCH_VERSION

```
public static final int MATCH_VERSION = 50
```

Constant for the USB device match scale, indicating a match with usbinfo.idVendor, usbinfo.idProduct and usbinfo.bcdDevice. Value is 50.

MATCH_MODEL

```
public static final int MATCH_MODEL = 40
```

Constant for the USB device match scale, indicating a match with usbinfo.idVendor and usbinfo.idProduct. Value is 40.

MATCH_PROTOCOL

```
public static final int MATCH_PROTOCOL = 30
```

Constant for the USB device match scale, indicating a match with usbinfo.bDeviceClass, usbinfo.bDeviceSubClass and usbinfo.bDeviceProtocol, or a match with bInterfaceClass, bInterfaceSubClass and bInterfaceProtocol in one of usbinfo.interfaceclasses. Value is 30.

MATCH_SUBCLASS

```
public static final int MATCH_SUBCLASS = 20
```

Constant for the USB device match scale, indicating a matchusbinfo.bDeviceClass and usbinfo.bDeviceSubClass, or a match with bInterfaceClass and bInterfaceSubClass in one of usbinfo.interfaceclasses. Value is 20.

MATCH_CLASS

```
public static final int MATCH_CLASS = 10
```

Constant for the USB device match scale, indicating a match with usbinfo.bDeviceClass, or a match with bInterfaceClass in one of usbinfo.interfaceclasses. Value is 10.

8 Considered Alternatives

8.1 usbinfo.interfaceclasses

The alternative format of usbinfo.interfaceclasses is below.

- `usbinfo.interfaceclasses` - MANDATORY property key. The property value is an array of `String` for each interface descriptor. Each `String` value is connected with "_" interface descriptor's *bInterfaceClass*, *bInterfaceSubClass*, and *bInterfaceProtocol*. If there is no subclass code associated with the class code, does not connect subclass code and protocol code. If there is no protocol code associated with the class code, the protocol code is not connected. In addition, if the class code is vendor-specific class, does not connect subclass code and protocol code. Set only the class code. (See Table 4.)
  - Example: An example of a USB device that has 2 interfaces. The first class code is CDC, subclass code is ACM(without protocol code). The second class code is CDC-Data (no subclass code and protocol code).
    - Value: "CDC_ACM", "CDC-Data"

Table 4: Class Code and Service Property

	Class Code	SubClass Code	Protocol Code	Representation in Service Property
1	01	01	any	Audio_AudioControl
2	01	02	any	Audio_AudioStreaming
3	01	03	any	Audio_MidiStreaming
4	01	other than above	any	Audio

5	02	01	01	CDC_DLCM_V.250
6	02	01	other than above	CDC_DLCM
7	02	02	01	CDC_ACM_V.250
8	02	02	other than above	CDC_ACM
9	02	03	01	CDC_TCM_V.250
10	02	03	other than above	CDC_TCM
11	02	04	any	CDC_MCCM
12	02	05	any	CDC_CAPI
13	02	06	any	CDC_ENCM
14	02	07	any	CDC_ATM
15	02	08	02	CDC_WHCM_PCCA-101
16	02	08	03	CDC_WHCM_PCCA-101-AnnexO
17	02	08	04	CDC_WHCM_GSM7.07
18	02	08	05	CDC_WHCM_3GPP27.007
19	02	08	06	CDC_WHCM_TIA-CDMA
20	02	08	FE	CDC_WHCM_ExternalProtocol
21	02	08	other than above	CDC_WHCM
22	02	09	02	CDC_DM_PCCA-101
23	02	09	03	CDC_DM_PCCA-101-AnnexO
24	02	09	04	CDC_DM_GSM7.07
25	02	09	05	CDC_DM_3GPP27.007
26	02	09	06	CDC_DM_TIA-CDMA
27	02	09	FE	CDC_DM_ExternalProtocol
28	02	09	other than above	CDC_DM
29	02	0A	02	CDC_MDLM_PCCA-101
30	02	0A	03	CDC_MDLM_PCCA-101-AnnexO
31	02	0A	04	CDC_MDLM_GSM7.07
32	02	0A	05	CDC_MDLM_3GPP27.007
33	02	0A	06	CDC_MDLM_TIA-CDMA
34	02	0A	FE	CDC_MDLM_ExternalProtocol
35	02	0A	other than above	CDC_MDLM
36	02	0B	02	CDC_OBEX_PCCA-101
37	02	0B	03	CDC_OBEX_PCCA-101-AnnexO
38	02	0B	04	CDC_OBEX_GSM7.07
39	02	0B	05	CDC_OBEX_3GPP27.007
40	02	0B	06	CDC_OBEX_TIA-CDMA
41	02	0B	FE	CDC_OBEX_ExternalProtocol
42	02	0B	other than above	CDC_OBEX
43	02	0C	07	CDC_EEM_EEM
44	02	0C	other than above	CDC_EEM
45	02	0D	FE	CDC_NCM_ExternalProtocol
46	02	0D	other than above	CDC_NCM
47	02	other than above	any	CDC
48	03	01	01	HID_Boot_KeyBoard
49	03	01	02	HID_Boot_Mouse
50	03	01	other than above	HID_Boot
51	03	other than above	any	HID
52	05	any	any	Physical
53	06	01	01	Image_Capture_PIMA15740
54	06	01	other than above	Image_Capture
55	06	other than above	any	Image
56	07	01	01	Printer_Printer_Unidirectional
57	07	01	02	Printer_Printer_Bi-directional
58	07	01	03	Printer_Printer_1284.4
59	07	01	FF	Printer_Printer_VendorSpecific
60	07	01	other than above	Printer_Printer
61	07	other than above	any	Printer
62	08	00	any	MassStorage_SCSI-nr
63	08	01	any	MassStorage_RBC
64	08	02	00	MassStorage_MMC-5_CBI-cci

65	08	02	01	MassStorage_MMC-5_CBI-ncci
66	08	02	50	MassStorage_MMC-5_BBB
67	08	02	other than above	MassStorage_MMC-5
68	08	04	00	MassStorage_UFI_CBI-cci
69	08	04	01	MassStorage_UFI_CBI-ncci
70	08	04	other than above	MassStorage_UFI
71	08	06	50	MassStorage_SCSI_BBB
72	08	06	62	MassStorage_SCSI_UAS
73	08	06	other than above	MassStorage_SCSI
74	08	07	any	MassStorage_LSDFS
75	08	08	any	MassStorage_IEEE1667
76	08	FF	00	MassStorage_VendorSpecific_CBI-cci
77	08	FF	01	MassStorage_VendorSpecific_CBI-ncci
78	08	FF	50	MassStorage_VendorSpecific_BBB
79	08	FF	other than above	MassStorage_VendorSpecific
80	08	other than above	any	MassStorage
81	0A	00	01	CDC-Data_Data_NTB
82	0A	00	30	CDC-Data_Data_I.430
83	0A	00	31	CDC-Data_Data_HDLC
84	0A	00	32	CDC-Data_Data_Transparent
85	0A	00	50	CDC-Data_Data_Q.921M
86	0A	00	51	CDC-Data_Data_Q.921
87	0A	00	52	CDC-Data_Data_Q.921TM
88	0A	00	90	CDC-Data_Data_V.42bis
89	0A	00	91	CDC-Data_Data_Euro-ISDN
90	0A	00	92	CDC-Data_Data_V.120
91	0A	00	93	CDC-Data_Data_CAPI2.0
92	0A	00	FD	CDC-Data_Data_HostBasedDriver
93	0A	00	FE	CDC-Data_Data_PUFD
94	0A	00	FF	CDC-Data_Data_VendorSpecific
95	0A	00	other than above	CDC-Data_Data
96	0A	other than above	any	CDC-Data
97	0B	00	00	SmartCard_SmartCard_CCID
98	0B	00	01	SmartCard_SmartCard_ICC-A
99	0B	00	02	SmartCard_SmartCard_ICC-B
100	0B	00	other than above	SmartCard_SmartCard
101	0B	other than above	any	SmartCard
102	0D	00	00	ContentSecurity
103	0E	01	any	Video_VideoControl
104	0E	02	any	Video_VideoStreaming
105	0E	03	any	Video_VideoInterfaceCollection
106	0E	other than above	any	Video
107	0F	any	any	PersonalHealthcareDevice
108	10	01	any	AudioVideoDevice_AVControlInterface
109	10	02	any	AudioVideoDevice_ AVDataVideoStreamingInterface
110	10	03	any	AudioVideoDevice_ AVDataAudioStreamingInterface
111	10	other than above	any	AudioVideoDevice
112	DC	any	any	DiagnosticDevice
113	E0	01	01	WirelessController_Wireless_Bluetooth
114	E0	01	02	WirelessController_Wireless_UWB
115	E0	01	03	WirelessController_Wireless_RemoteNDIS
116	E0	01	04	WirelessController_Wireless_ BluetoothAMPController
117	E0	01	other than above	WirelessController_Wireless
118	E0	02	01	WirelessController_WireAdapter_Host
119	E0	02	02	WirelessController_WireAdapter_Device
120	E0	02	03	WirelessController_WireAdapter_ DeviceIsochronous

121	E0	02	other than above	WirelessController_WireAdapter
122	E0	other than above	any	WirelessController
123	EF	01	any	Miscellaneous_Sync
124	EF	03	any	Miscellaneous_CBAF
125	EF	other than above	any	Miscellaneous
126	FE	01	any	ApplicationSpecific_FirmwareUpgrade
127	FE	02	any	ApplicationSpecific_IrdaBridge
128	FE	03	01	ApplicationSpecific_TestMeasurement_ USB488
129	FE	03	other than above	ApplicationSpecific_TestMeasurement
130	FE	other than above	any	ApplicationSpecific
131	FF	any	any	VendorSpecific

9 Security Considerations

ServicePermission is needed when a bundle get USBInfoDevice service.

10 Document Support

10.1 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]. Universal Serial Bus Specification Revision 1.1, September 23, 1998.

10.2 Author's Address

Name	Yukio KOIKE
Company	NTT Corporation
Address	1-1, Hikari-no-oka, Yokosuka-shi, 238-0847, Kanagawa, Japan
Voice	+81 46 859 5142
e-mail	koike.yukio@lab.ntt.co.jp

10.3 Acronyms and Abbreviations

10.4 End of Document