MCPC
モバイルコンピューティング推進コンソーシアム
Mobile Computing Promotion Consortium

MCPC TR-019

Android 対応

USB 機器開発ガイドライン

Version 1.00

2014 年 1 月 30 日

モバイルコンピューティング推進コンソーシアム
技術委員会

変更履歴

日付	Version	変更内容
2014年1月30日	1.00	Base version initial release.

ドキュメント発行者、および著作権者:
〒105-0011
東京都港区芝公園3-5-12 芝公園真田ビル
モバイルコンピューティング推進コンソーシアム (MCPC)
電話：03-5401-1935
ＦＡＸ：03-5401-1937
ＥＭＡＩＬ：office@mcpc-jp.org
WEB SITE: http://www.mcpc-jp.org

機密保持について:
MCPC会則、IP Policyを遵守する。

免責について:
本ドキュメントはモバイルコンピューティングに関する標準仕様、推奨仕様などを提供するもので、モバイルコンピューティング推進コンソーシアム（以下MCPCとする）は、本ドキュメントを使用した結果発生した損害、第三者の特許、またはその他の権利の侵害に対して、一切の責任を負わない。また、本ドキュメントはMCPC、または第三者が保持するいかなる権利のライセンスを許諾するものではない。

2進数、10進数、16進数の表記方法：
2進数は小文字”b”を付加する。(例：10b)
2進数4桁以上は4桁ごとにスペースで区切る。(例：1000 0101 0010b)
16進数は小文字”h”を付加する。(例：FFFFh and 80h)
その他の数字表記は10進数とする。

キーワード

することができる してもよい (may)	推奨または要求に自由な選択肢を示す。
すべきである (should)	必須ではないが強い推奨を示す。実施の際、必須ではないが考慮すべき。
しなければならない (shall)	必須要求を示す。接続性、仕様準拠のために必ず実施しなければならない。

Table of Contents

Table of Figures

1. はじめに

本書は、Android OSを搭載したスマートフォンにUSB接続の外部機器を開発するに当たり、外部機器、アプリケーション及びスマートフォンの各開発者に対して、開発上の注意点などをまとめたものである。
本書の中で記載されているUSB Host APIおよびUSB Accessoryモードは、Google社が提供するAndroid OSの機能である。本書は、その仕様に基づき、その利用方法について規定する。

※USB Host APIおよびUSB Accessoryモードの詳細については、Android Developersガイドを参照のこと。

http://developer.android.com/guide/topics/connectivity/usb/index.html

2. Android環境でのUSB機能概要

2.1. Android端末のUSBインタフェース環境

(1)従来アーキテクチャー:

① 従来より、Android 端末を USB メモリ等の USB アクセサリ機器として PC に接続することが可能である。
これは、予めAndroidのカーネル層 (Linux) に各デバイスドライバを組み込んでおき、Androidからファンクションモードを切り替える (VID/PIDを切り替える) ことで、いろいろなファンクションのアクセサリとしてPCに接続することが可能となる。
※PCとの接続は、本接続方式となる。

②USBホスト機能を有するAndroid端末では、一部の外部USB機器と接続して使用することが可能である。
これは、予めAndroidのカーネル層 (Linux) に各ファンクションのホストドライバを組み込んでおき、各ファンクションの外部USB機器と接続することで、Android端末がUSBホストとして外部USB機器を使用することができる。

(2)Androidの新アーキテクチャー:

・Androidでは、USB外部機器との接続のために、上記の従来アーキテクチャーに加えて、ドライバの組み込みを必要とせず、ユーザ層 (java) であるアプリからUSB制御を行うための仕組みが提供されている。

①USB Accessoryモード:
・外部機器としてUSBホスト機能を有する非PCを対象とする。Android端末はUSBデバイスとして接続するが、外部機器の制御はAndroid端末側から行うモードである。

②Host APIモード:
・市販の外部USB機器との接続を目的として、ドライバの組み込みを必要とせず、Android端末をUSBホストとして接続するためのモードである。

- カーネル(Linux)層の USB ライブラリからアクセスすると、「ioctl」で USB 転送ドライバが起動され、データ転送が行われる。

2.2. Android端末のUSBインタフェース環境

2.2.1 PC(Windows OS)でのUSBホスト/デバイスのトポロジの考え方

(1) PC(Windows OS)のドライバインストール方法

・クラス標準ドライバの多くは、Windows のINBOXドライバとして予め組み込まれているため、ユーザによるドライバインストールは不要である。
・USB外部機器個別のドライバは、USB外部機器と一緒に提供され、Windows OSのドライバインストールのフレームワークにより、ユーザが外部記憶装置よりドライバをインストールする。

※後からドライバをインストールする仕組みは、Windows OS固有の技術

2.2.2 Windows OSでのUSBハンドリングモデル

OSのソフトウェアスタックとユーザソフトウェアの階層、USB-IFのデバイスクラスの関係をFigure 2-1に示す。

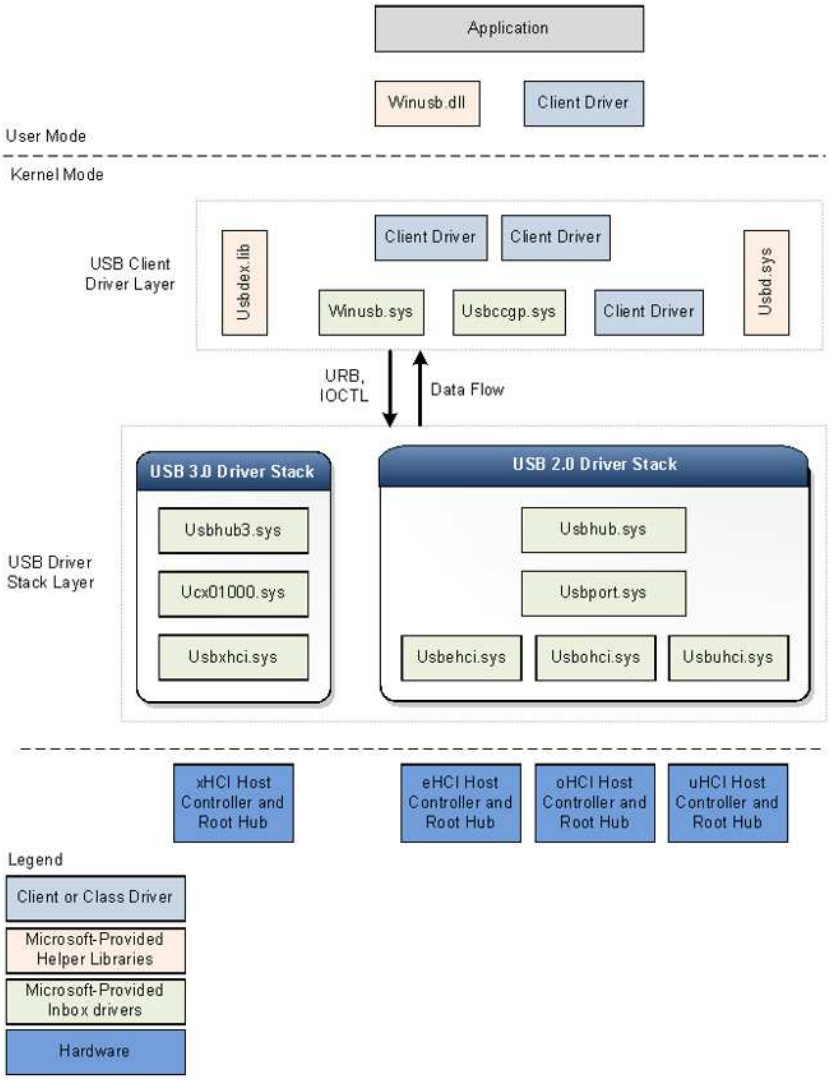


Figure 2-1 windows OS での USB ハンドリングモデル

2.2.3 Android端末でのUSBホスト/デバイスのトポロジの考え方

・Androidではカーネル層が構成されている。ここにUSBドライバが組み込まれるが、PC(Windows OS)と異なる以下の制約がある。

[Android 端末の制約]
1) 個別のUSBドライバをユーザが後からインストールする仕組みがない。
2) ドライバを組み込むためには、Linuxカーネルソースを Buildする必要がある。
（ルート権限を使う等の特殊な方法を除く）

(1)Android端末の既存アーキテクチャーにおける一般的なドライバの組み込み方法
・USBホスト／デバイス側ともに、予め実装するクラス標準ドライバ（Linuxのオープンソース）および個別ドライバを用意して、ソースをBuildして組み込む。

※Androidの既存アーキテクチャーでは、ユーザが後からドライバをインストールする仕組みがない。

この問題に対する解決策の1つとして、USB Accessoryモード / Host APIモードが提供された。

2.2.4 Android OSでの既存USBハンドリングモデル

OSのソフトウェアスタックとユーザソフトウェアの階層、USB-IFのデバイスクラスの関係をFigure 2-2に示す。

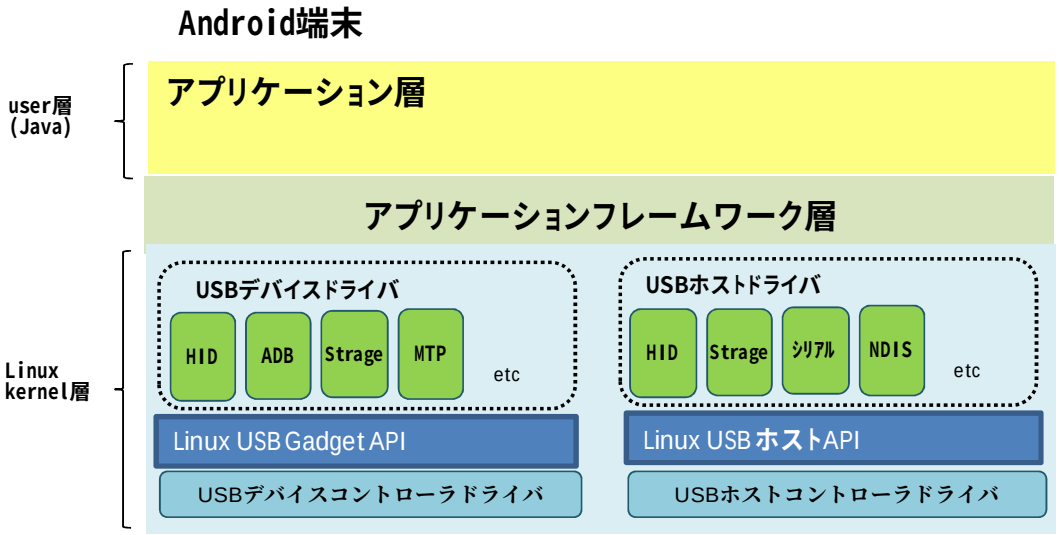


Figure 2-2 Android OS での既存 USB ハンドリングモデル

※上記、Linux Kernel層の「USBデバイスドライバ」および「USBホストドライバ」を組み込むには、Buildの必要がある。

2.3. USB Host APIモード概要

2.3.1 Host API モードでの USB 機器の接続モデル

(1) USB Host API モード

・Android 端末側が USB Host になり、Java アプリ層で制御する。

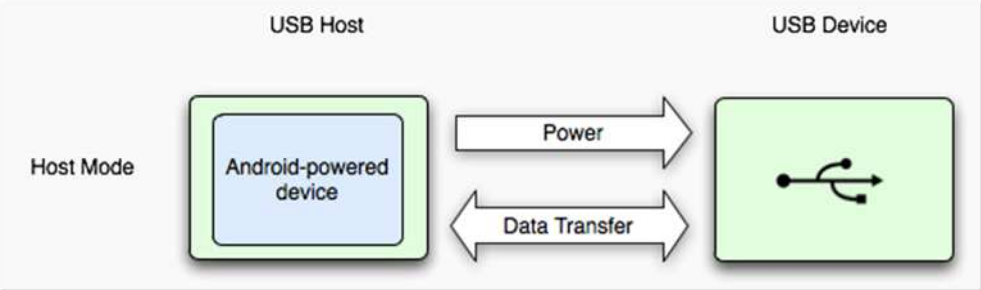


Figure 2-3 USB Host API モード

2.3.2 Host API モードでの USB 機器の実装モデル

2.3.2.1. モバイル端末での実装

(1)構成

・USB Host APIインタフェース仕様に準拠したユーザ層 (Java) アプリを提供するだけで、USB外部機器とのデータ通信が可能となる。

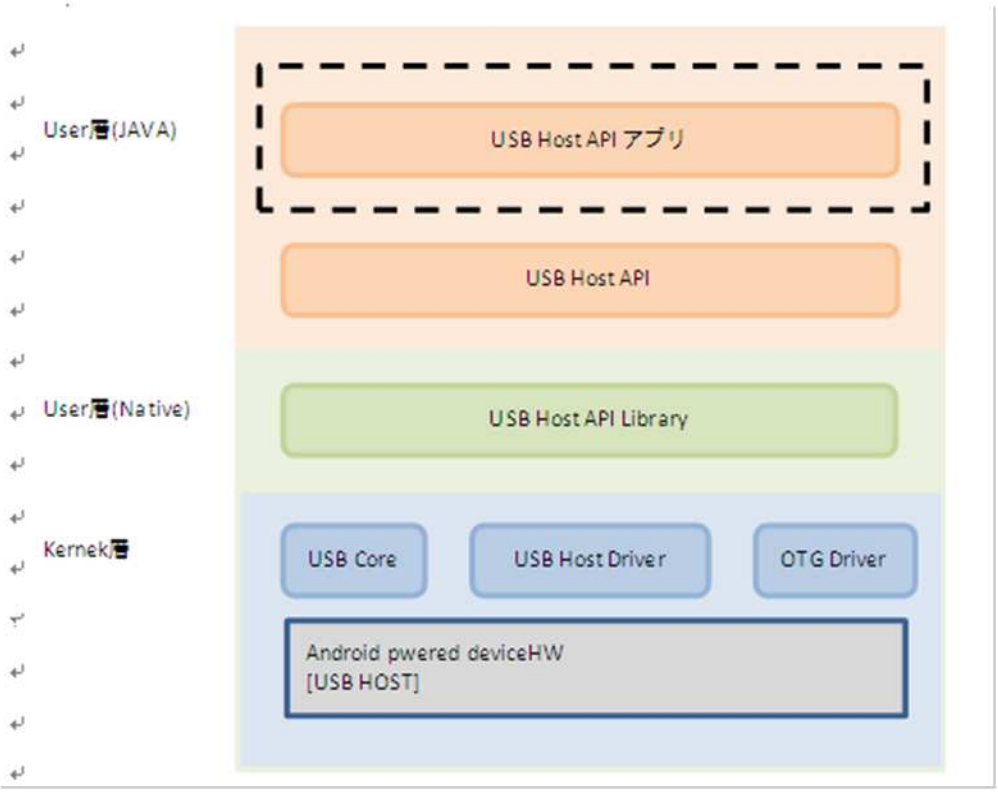


Figure 2-4 Host API モードでの USB 機器実装モデル

(2) 概要

①Android 3.1以降でサポート
②Android端末がUSBホストとして動作する。
③Android端末側からUSB外部デバイスに、USB VBusの5Vを供給する。
④USBドライバはAndroidの標準ドライバが使用される。
⑤Isochronous転送は非サポート。(Interrupt転送はサポート)
⇒音声や映像の標準インタフェースに対応したUSB外部機器は接続不可。
⑥USBインターフェースおよびエンドポイント構成は、外部USB機器で実装されている構成をAndroid端末のJavaアプリで設定する。
⑦Javaアプリは、USB Host APIで定義されたAPIを使用して、USB外部デバイスの検出/通信ポートオープン/データ転送の処理を行う。
⑧外部デバイス側およびAndroid端末側に対して、特別なプロトコル対応は不要。
⑨Android4.0では、CTS (Compatibility Test Suite)で必須要件となっているため、Android4.0端末ではHost APIの対応が必須。

(3) 特徴

①必要とするUSBドライバをカーネル層に組み込むことなく、ユーザ層 (Java) アプリは、Android USB Host APIからUSB上を流れる制御データおよびユーザデータの送受信を行う。

②Android USB Host APIから受信した制御データおよびユーザデータを処理するファンクション部分は、ユーザ層 (Java) アプリで処理する必要がある。

③上記処理には、Android 端末機種に依存する部分はないため、ユーザ層 (Java) アプリは、Android 端末機種共通のアプリとして提供可能である。

④USB 外部機器側には、特別な対応や修正は不要であり、Windows PC で使用している市販の USB 外部機器をそのまま接続することができる。
(ただし、Isochronous転送を使用したUVC(Video)機器やUAC (Audio)機器は接続不可)

2.3.2.2. USB外部デバイスでの実装

・特別な変更の必要がないため、市販PC　Windows用USB外部デバイスがそのまま使用可能。

2.4. USB Accessoryモード概要

2.4.1 USB AccessoryモードでのUSB機器の接続モデル

(1) USB Accessory モード
・外部 USB 機器が USB Host になり、Android 端末が USB Device として動作する。

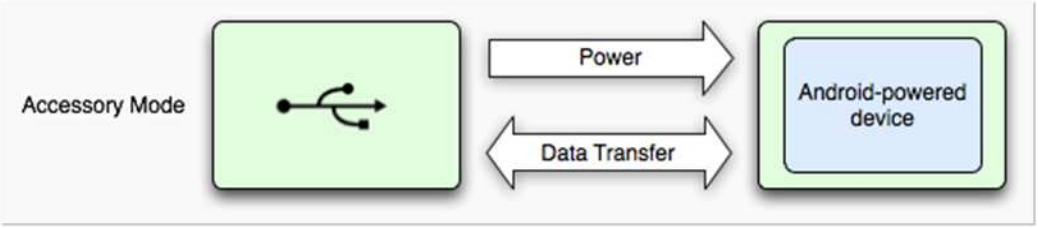


Figure 2-5 USB Accessory モードでの USB 接続モデル

2.4.2 USB AccessoryモードでのUSB機器の実装モデル

2.4.2.1 モバイル端末での実装

(1)構成
・Android端末側は、USB Accessoryモード仕様に準拠したユーザ層 (Java) アプリを提供する。
・USB外部デバイス側は、USBホスト機能およびOpen Accessoryプロトコル仕様に準拠したスタックを実装する必要がある。

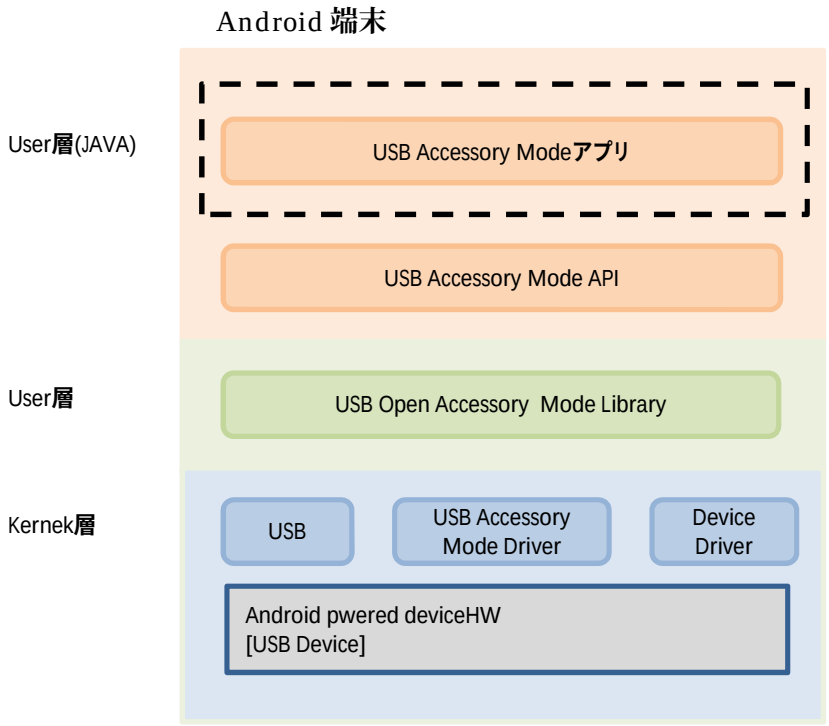


Figure 2-6 モバイル端末での USB Accessory モードの実装

(2) 概要

①Android 2.3.4以降でサポート
②Android端末がUSBデバイスとして動作する。
③外部USBデバイス側からUSB VBusを供給する。
④外部USBデバイス側はUSBホスト機能を持つ。
⑤使用できるPID/VIDおよびそれに対応したUSBインターフェースおよびエンドポイント構成は、Open Accessory Protocolで規定された固定の構成となる。
⑥外部USBデバイス側は、Android Open Accessory (AOA) Protocolに対応している必要がある。
⑦Javaアプリは、USB Accessory APIで定義されたAPIを使用して、USB外部デバイスの検出/通信ポートオープン/データ転送の処理を行う。
⑧Android4.0では、CTS (Compatibility Test Suite)で必須要件となっているため、Android4.0端末では対応が必須。
⑨Android4.1のCTSではAudio Classインタフェースの対応が必須。

(3) 特徴

①USB外部機器がUSBホストとなるため、Android端末側はUSBから充電することが可能である。

②USB外部機器は、Android Open Accessory Protocolで規定されたUSBインタフェース構成とし、Android Open Accessory Protocolに対応する必要がある。
このため、USB外部機器はWindows PC用のUSB外部機器とは異なり、Android端末専用外部機器となる。

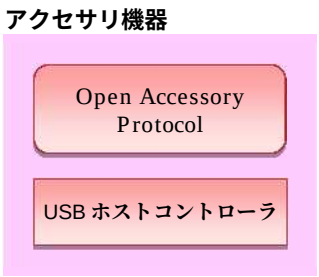
③Android 端末は、Android Open Accessory Protocol は標準実装されている。ユーザ層 (Java) アプリは、この Android Open Accessory Protocol を使って、USB 外部機器の制御およびユーザデータの送受信処理を行う。

④上記処理には、Android 端末機種に依存する部分はないため、ユーザ層 (Java) アプリは、Android 端末機種共通のアプリとして提供可能である。

⑤AOA2.0では、Audio Class Interfaceがサポートされているため、Android端末の音楽コンテンツをスピーカに出力するための外部機器が想定されている。

2.4.2.2 USB外部デバイスでの実装

・USB外部デバイスは、以下の機能が必要となる。
1) USB ホストコントローラ
2) Android Open Accessary Protocol



※既存PC Windows用USB外部デバイスとは互換性がない。

Figure 2-7 USB 外部デバイスでの実装

3. Android環境でのUSBアプリインタフェース

3.1. USB Host APIモードAPI概要

(1)android.hardware.usb class仕様

・USB Host APIとUSB Accessory ModeのAPIが共通定義されている。

クラス	説明
UsbManager	USB Host API の中心クラス。本クラスを介してデバイス情報を列挙する。 USB の状態をアクセスし、接続された外部機器との通信を行うためのクラス
UsbDevice	USB で接続されているデバイス。UsbManager が列挙するデバイス情報に含まれる。 Android 端末が USB ホストとして動作する時に、接続されている外部機器と通信するためのクラス。
UsbDeviceConnection	対象のデバイスとの間のデータ（バルク転送等）および USB コントロールメッセージを送受信するためのクラス。
UsbEndpoint	USB を介してデータを送受信する際の UsbInterface 上の Endpoint。 Endpoint0 はホストからデバイスへ制御コマンドを送信する USB コントロールメッセージ用の Endpoint。（isochronous 転送は未サポート）
UsbInterface	USB デバイスのインターフェース。デバイスごとに 1 つ以上のインターフェースを持つ。データを送受信するときは、適切なインターフェースのエンドポイントを利用する必要がある。
UsbRequest	UsbDeviceConnection を介したデバイスと通信するための非同期リクエストを表現する。 UsbRequest.queue()を使用して、データ転送要求を行い、UsbRequest を UsbDeviceConnection.requestWait()で転送完了要求を待ち合わせる。 Endpoint0(制御コマンド送信)は、本クラス利用は未サポート。
UsbConstants	USB の制御に用いる定数定義。デバイスのクラス(オーディオとかストレージ等)の定義をしている。
UsbAccessory	USB をアクセサリ(デバイス)モードで接続するクラス。 端末がアクセサリとなり、ホストと通信する際に利用する。

(2) android.hardware.usb API仕様

※各APIの詳細については、下記URLのAndroid Developers'ガイドを参照のこと。

①UsbManager

・接続されたUSBデバイスリストを取得する。
(http://developer.android.com/reference/android/hardware/usb/UsbManager.html)

API	概要
getDeviceList	接続されている USB デバイスのリストを取得する。
requestPermission	デバイスに対して、権限の設定を要求する。 (ユーザにダイアログ表示して許可を求める)
hasPermission	デバイスに対する権限の返却値を取得
openDevice	デバイスの Open を行う。

②UsbDevice

・プロダクト ID/ベンダ ID/クラス ID 等を取得する。
(http://developer.android.com/reference/android/hardware/usb/UsbDevice.html)

API	概要
describeContents	Parcelable な整列化表現された専用オブジェクトの種類を記述する。
equals	指定されたオブジェクトとこのインスタンスを比較し、等しいかどうかを示す。
getDeviceClass	デバイスクラスの値を取得
grtDeviceid	デバイス ID を取得する。
getDeviceName	デバイス名を取得する。
getDeviceProtocol	デバイスプロトコルの値を取得する。
getDeviceSubclass	デバイスサブクラスの値を取得する。
getInterface	指定した Index 番号の UsbInterface を取得する。
getInterfaceCount	UsbInterface の数を取得する。
getProductId	プロダクト ID を取得する。
getVendorId	ベンダーID を取得する
hashCode	このオブジェクトのハッシュコードを取得する。
toString	ストリング内容を取得する。
writeToParcel	このオブジェクトを Parcel にする。

③UsbDeviceConnection

(http://developer.android.com/reference/android/hardware/usb/UsbDeviceConnection.html)

API	概要
bulkTransfer	UsbEndpoint で指定された Endpoint で bulk 転送を実行する。
claimInterface	UsbInterface のオブジェクトを取得 (生成) する。
close	UsbManger::OpenDevice で Open したデバイスを Close する。
controlTransfer	デバイスに対して、endpoint0 の control 転送を行う。
getFileDescriptor	デバイスのファイルディスクリプタを取得する。
getRawDescriptor	デバイスの生のディスクリプタを取得する。
getSerial	デバイスのシリアル番号を取得する。
releaseInterface	UsbInterface のオブジェクトを解放する。
requestWait	データ受信要求(UsbRequest::Queue)の待ち合わせを行う。

⑤UsbEndpoint

(http://developer.android.com/reference/android/hardware/usb/UsbEndpoint.html)

API	概要
describeContents	Parcelable な整列化表現された専用オブジェクトの種類を記述する。
getAddress	Endpoint のアドレスを取得する。
getAttributes	Endpoint の属性情報を取得する。
getDirection	Endpoint の方向を取得する。
getEndpointNumber	Endpoint 数を取得する。
getInterval	Endpoint の転送間隔を取得する。
getMaxPacketSize	Endpoint の最大パケットサイズを取得する。
getType	Endpoint タイプを取得する。
toString	ストリング内容を取得する。
writeToParcel	このオブジェクトを Parcel にする。

⑥UsbInterface

(http://developer.android.com/reference/android/hardware/usb/UsbInterface.html)

API	概要
describeContents	Parcelable な整列化表現された専用オブジェクトの種類を記述する。
getEndpoint	UsbEndpoint を取得する。
getEndpointCount	UsbEndpoint の数を取得する。
getId	インターフェース ID を取得する。
getInterfaceClass	インターフェースクラスを取得する。
getInterfaceProtocol	インタフェースプロトコルを取得する。
getInterfaceSubclass	インターフェースサブクラスを取得する。
toString	ストリング内容を取得する。
writeToParcel	このオブジェクトを Parcel にする。

⑦UsbConstants

(http://developer.android.com/reference/android/hardware/usb/UsbConstants.html)

・USB 定数の定義値が提供されている。 (linux/usb/ch9.h)

⑧UsbRequest

(http://developer.android.com/reference/android/hardware/usb/UsbRequest.html)

API	概要
cancel	キューイングのキャンセルを行う。
close	このリクエストに関連する全てのリソースを解放する。
getClientData	このリクエストに対するクライアントデータを取得する。
getEndpoint	このリクエストに対する UsbEndpoint を取得する。
Initialize	エンドポイントの初期化を行う。
Queue	エンドポイントに対して Read/Write の要求を行う。
setClientData	そのリクエストに対するクライアントデータを設定する。

3.2. USB AccessoryモードAPI概要

(1)android.hardware.usb class仕様

・USB Host APIとUSB Accessory ModeのAPIが共通定義されている。
(3.1章(1)参照)

(2) android.hardware.usb API仕様

※各APIの詳細については、下記URLのAndroid Developers'ガイドを参照のこと。

①UsbManager

・接続されたUSBデバイスリストを取得する。
(http://developer.android.com/reference/android/hardware/usb/UsbManager.html)

API	概要
getAccessoryList	接続されている USB アクセサリのリストを取得する。
requestPermission	アクセサリに対して、権限の設定を要求する。 (ユーザにダイアログ表示して許可を求める)
hasPermission	アクセサリに対する権限の返却値を取得
openAccessory	アクセサリの Open を行う。

②UsbDevice

※USB Accessory Mode では使用しない。

③UsbDeviceConnection

(http://developer.android.com/reference/android/hardware/usb/UsbDeviceConnection.html)

API	概要
bulkTransfer	UsbEndpoint で指定された Endpoint で bulk 転送を実行する。
claimInterface	UsbInterface のオブジェクトを取得 (生成) する。
close	UsbManger::OpenDevice で Open したデバイスを Close する。
controlTransfer	デバイスに対して、endpoint0 の control 転送を行う。
getFileDescriptor	デバイスのファイルディスクリプタを取得する。
getRawDescriptor	デバイスの生のディスクリプタを取得する。
getSerial	デバイスのシリアル番号を取得する。
releaseInterface	UsbInterface のオブジェクトを解放する。
requestWait	データ受信要求(UsbRequest::Queue)の待ち合わせを行う。

④UsbEndpoint

(http://developer.android.com/reference/android/hardware/usb/UsbEndpoint.html)

API	概要
describeContents	Parcelable な整列化表現された専用オブジェクトの種類を記述する。
getAddress	Endpoint のアドレスを取得する。
getAttributes	Endpoint の属性情報を取得する。
getDirection	Endpoint の方向を取得する。
getEndpointNumber	Endpoint 数を取得する。
getInterval	Endpoint の転送間隔を取得する。
getMaxPacketSize	Endpoint の最大パケットサイズを取得する。
getType	Endpoint タイプを取得する。
toString	ストリング内容を取得する。
writeToParcel	このオブジェクトを Parcel にする。

⑤UsbInterface

(http://developer.android.com/reference/android/hardware/usb/UsbInterface.html)

API	概要
describeContents	Parcelable な整列化表現された専用オブジェクトの種類を記述する。
getEndpoint	UsbEndpoint を取得する。
getEndpointCount	UsbEndpoint の数を取得する。
getId	インターフェース ID を取得する。
getInterfaceClass	インターフェースクラスを取得する。
getInterfaceProtocol	インタフェースプロトコルを取得する。
getInterfaceSubclass	インターフェースサブクラスを取得する。
toString	ストリング内容を取得する。
writeToParcel	このオブジェクトを Parcel にする。

⑥UsbConstants

(http://developer.android.com/reference/android/hardware/usb/UsbConstants.html)

・USB 定数の定義値が提供されている。 (linux/usb/ch9.h)

⑦UsbRequest

(http://developer.android.com/reference/android/hardware/usb/UsbRequest.html)

API	概要
cancel	キューイングのキャンセルを行う。
close	このリクエストに関連する全てのリソースを解放する。
getClientData	このリクエストに対するクライアントデータを取得する。
getEndpoint	このリクエストに対する UsbEndpoint を取得する。
Initialize	エンドポイントの初期化を行う。
Queue	エンドポイントに対して Read/Write の要求を行う。
setClientData	そのリクエストに対するクライアントデータを設定する。

⑧UsbAccessory

(http://developer.android.com/reference/android/hardware/usb/UsbAccessory.html)

・USB Accessoryモードの接続形態の場合のみ使用する。

API	概要
describeContents	Parcelable な整列化表現された専用オブジェクトの種類を記述する。
equals	指定されたオブジェクトとこのインスタンスを比較し、等しいかどうかを示す。
getDescription	アクセサリの description を取得
grtManufacturer	アクセサリの製造者名を取得する。
getModel	アクセサリのモデル名を取得する。
getSerial	アクセサリのシリアル番号を取得する。
getUri	アクセサリの URI を取得する。
getVersion	アクセサリの版数を取得する。
hashCode	このオブジェクトのハッシュコードを取得する。
toString	ストリング内容を取得する。
writeToParcel	このオブジェクトを Parcel にする。

3.3. Android Open Accessory (AOA) Protocol

3.3.1 Android Open Accessory Protocol1.0 (AOA1.0)

①control転送のリクエストコマンド"51"の"Get Protocol"を外部機器からスマホへ送信し、スマホの対応により応答する値を変える。

0 ： AOA非対応
1 ： AOA1.0対応
2 ： AOA2.0対応

②control転送のリクエストコマンド"52"を外部機器からスマホへ送信し、stringディスクリプタとしてスマホから、適正なアプリがあるかどうかの情報をもらう。(数値をstring ID値で指定)

manufacturer name:	0
model name:	1
description:	2
version:	3
URI:	4
serial number:	5

③control転送のリクエストコマンド"53"を外部機器からスマホへ送信し、外部機器側から、AOAの開始を要求する。

④下記のAOA用のVID/PIDに切り換えるため、USBのバスリセットを行う。

VIDはGoogleの(0x18D1)

＜Accessoryデータ通信＞

product ID of 0x2D00： インタフェース=1つ 、BULKエンドポイント(IN/OUT) の2つ

＜Accessoryデータ通信＋ADB＞

product ID of 0x2D01： インタフェース=2つ 、BULKエンドポイント(IN/OUT) の2つ x2

3.3.2 Android Open Accessory Protocol2.0 (AOA2.0)

①AOA2.0で追加になったデバイス構成を以下に示す。
＜Audio通信＞
product ID of 0x2D02： 標準Audioクラスインタフェース

＜Audio通信＋ADB＞
product ID of 0x2D03： 標準Audioクラスインタフェース+ Bulk (IN/OUT) インタフェース

＜Accessoryデータ通信＋Audio通信＞
product ID of0x2D04： 標準Audioクラスインタフェース+ Bulk (IN/OUT) インタフェース

＜Accessoryデータ通信＋Audio通信＋ADB＞
product ID of0x2D05： 標準Audioクラスインタフェース+ Bulk (IN/OUT) インタフェース×2

②control転送のリクエストコマンド"58"を外部機器からスマホへ送信し、Audioクラスの設定値を外部機器からスマホへ送信する。
0 ： 無音 (デフォルト)
1 ： 2 channel, 16-bit PCM at 44100 Hz

③現状のAOAでは、スマホ⇒外部機器方向の音声しか対応していない。

④HID (Human Interface Device) の対応が必須となっている。
⇒外部機器側のスイッチから、スマホンのアプリを制御する。

⑤AOAでは、通常のHIDと異なり、外部機器側からスマートフォンにデータを通知する形になるが、これをcontrol転送で行う仕様を規定している。
(外部機器からAndroid端末の音楽コンテンツの選曲やボリューム調整等)

⇒HID用のエンドポイントやインタフェースは不要。

⑥HID用のリクエストコマンドが"54"～"57"で定義されている。

3.3.3 USB Accessoryモード切り替え

・USB アクセサリ機器側より、USB Accessoryモードへの切り替え指示 (get_protocol/プロトコルバージョン) を送出し、その指示によりAndroid端末側は、USB Accessoryモードに切り替えて、USB再接続を行う。

4. Android端末におけるUSBドライバ実装の推奨

4.1. USBデバイスドライバ（Linuxカーネル）実装

・PCとAndroid端末を接続した場合のユースケース（Android端末はUSBデバイス側となる）
※Android のUSBガジェットとして規定されている。

クラスドライバ	概要
MTP	Windows Media PlayerでPCとスマホのメディア（静止画、動画、音楽）を転送
マスストレージ	Android端末がUSBメモリとなり、PCとの間でコンテンツファイルを転送
RNDIS	Android端末を外付けモデムとして、PCからのインターネット接続（「テザリング」）
ADB	Androidアプリ開発用のポート
個別ドライバ	開発用およびメンテナンス用独自機能

4.2. USBホストドライバ（Linuxカーネル）実装

・外部USBデバイスとAndroid端末とを接続するユースケース（Android端末はUSBホスト側）
下記に、カーネル層（Linux）で標準提供されるクラスドライバの一例を示す。
※プラットホームや端末メーカの実装（Android端末のミドル層以上の実装）によっては使用できない場合や動作保障されていない場合がある。

クラスドライバ	概要
HID	キーボード、マウス、スタイラスペン等
マスストレージ	USBメモリ、HDD , DVD/CDドライブ等
RNDIS	LANアダプタ等
UVC	USB Video Class対応のWebカメラ
UAC	USB Audio Class対応のオーディオ機器

4.3. USB Host APIによる実装

・上記4.2章のAndroid端末の実装によらず、共通のアプリを提供することで、外部USB機器を使用することが可能となる。
※ただし、Android USB Host APIの制約により、Isochronous転送を使用したUVCやUAC等のストリーム転送は不可。

USB機器	概要
シリアルインタフェースを持つUSB機器	Windows PC上でCOMポートインタフェースとして見せる各種センサー機器等
Bulk転送を使った独自インタフェースのUSB機器	独自ドライバによる独自インタフェースのUSB機器

4.4. USB Accessoryモードによる実装

・USB Accessoryモード対応機器とAndroid端末との接続におけるユースケース
（Android端末はUSBデバイス側）

USB Accessory	概要
オーディオ機器	スマホの音楽コンテンツを再生する外部オーディオ機器。（外部機器側のスイッチやボリュームつまみで制御）
カーオーディオ機器	スマホの音楽コンテンツを再生する外部オーディオ機器。（外部機器側のスイッチやボリュームつまみで制御）

5. Android対応USBデバイスを作成する時の注意点

5.1. ハードウェア要件

(1) USB Host API モード

【USB外部機器】

① 市販の PC 用 USB デバイス機器 （但し UVC や UAC 等の Isochronous 転送を使用するものを除く）

【Android端末】

① USB ホストコントローラ機能
② USB からの電力供給

(2) USB Accessory モード

【USB外部機器】

① USB ホストコントローラ機能
② USB からの電力供給

【Android端末】

① USB デバイスコントローラ機能

5.2. ソフトウェア要件

(1) USB Host API モード

【USB外部機器】

（各デバイスのファンクション処理）

【Android端末】

① Android 3.1 以降の OS
② Android Host API および USB 機器のファンクション処理を含む Java アプリ

(2) USB Accessory モード

【USB外部機器】

① Android Open Accessory Protocol 機能

【Android端末】

① Android 2.3.4 以降の OS
② Android Open Accessory Protocol 処理を含む Java アプリ

6. 実装例

6.1. USB Accessoryモード実装例（Audio Player）

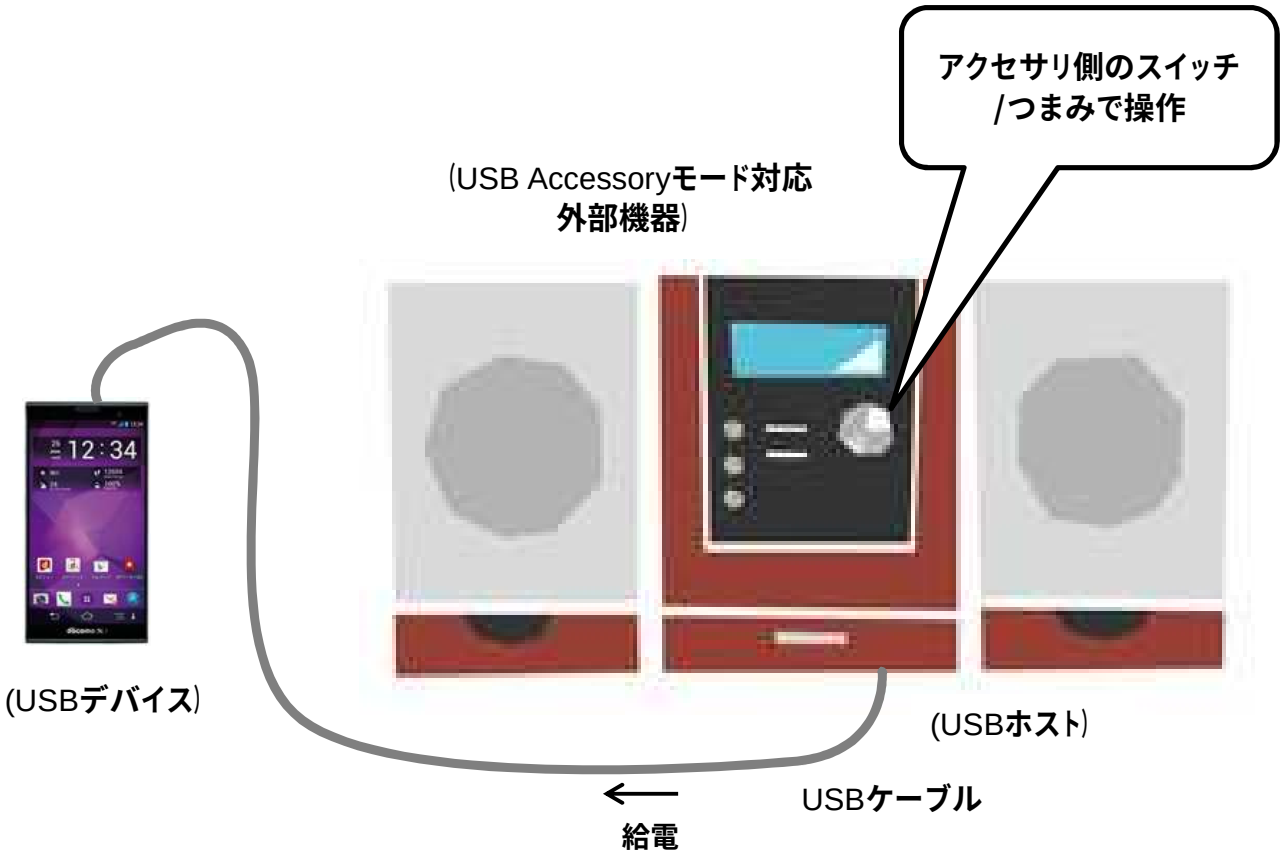


Figure 6-1 USB Accessory モード実装例(Audio Player)

6.1.1. USB Accessoryモード機器のUSBセットアップシーケンス

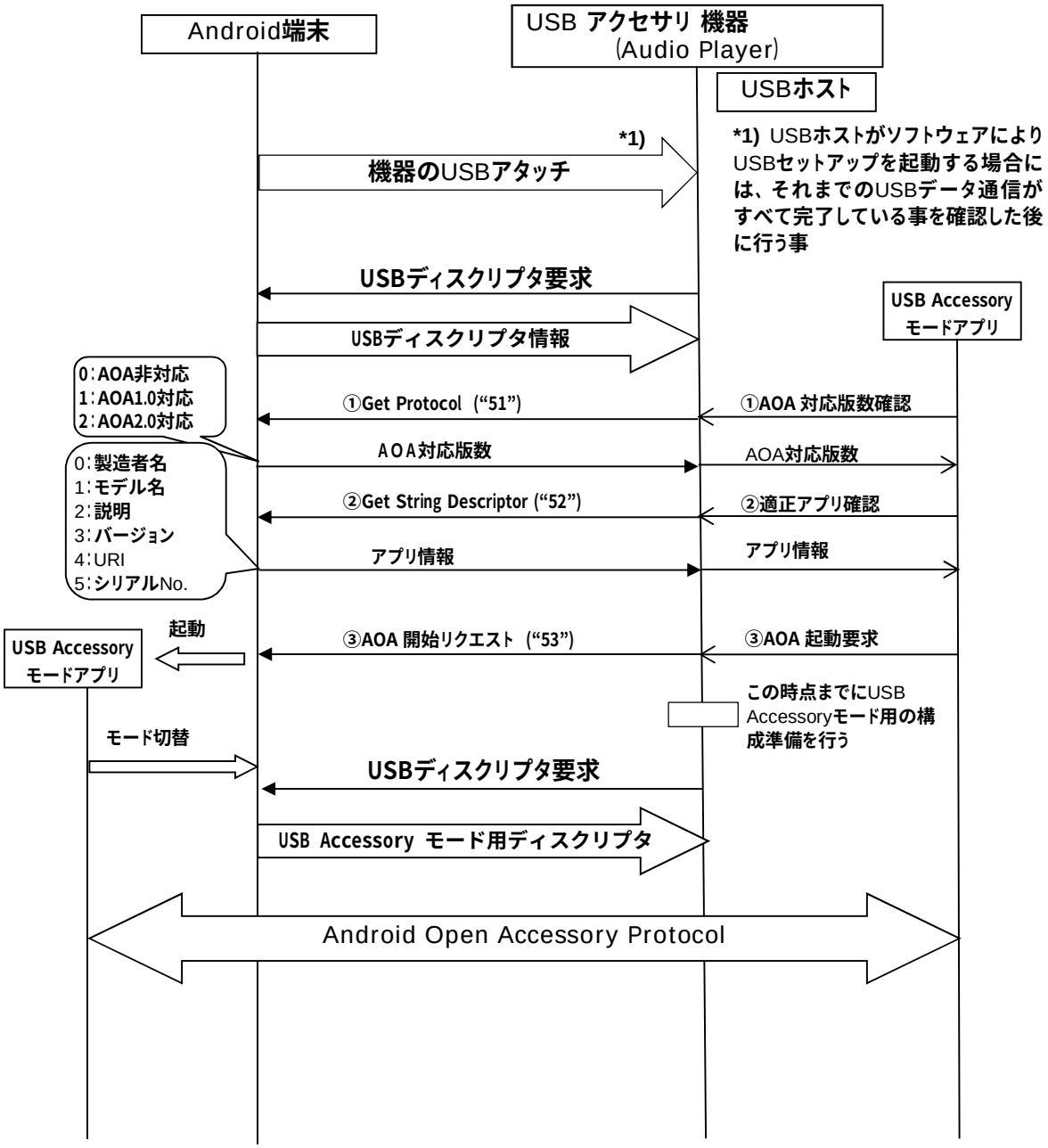


Figure 6-2 USB Accessory モード機器接続シーケンス

6.2. Android端末と車載器接続の実装考察

- **車載器側は、**Mirror Link **と** USB Accessory **モードの両方に対応し、スマホ側の** Mirror Link **対応有無により、モードを切り替える場合の動作を考察する。**

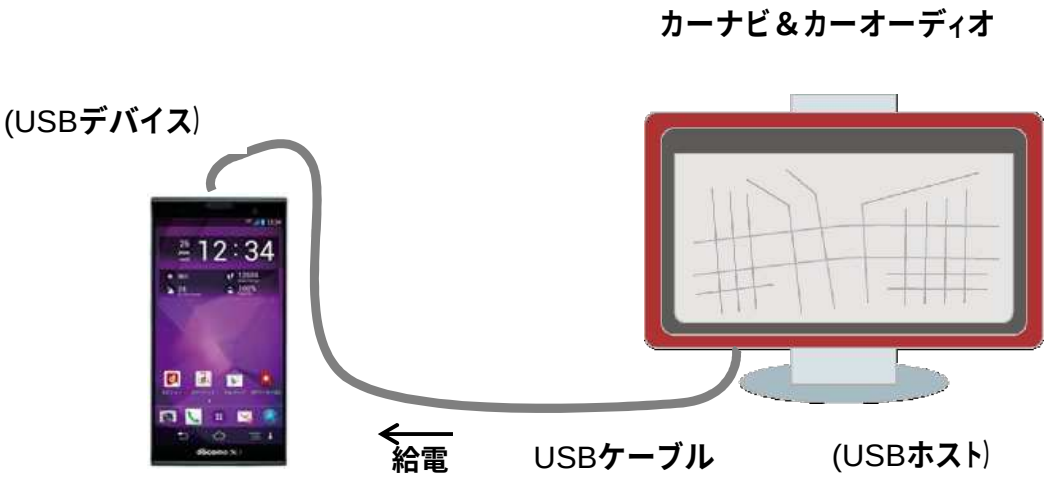


Figure 6-3 USB Accessory モード実装例(車載機)

① **車載器側は、スマホ側が** Mirror Link **モード可能かを確認する。(詳細は** MirrorLink **仕様書を参照のこと)**
② Mirror Link **モード不可の場合は、Figure 6-2 により、**USB Accessory **モードに切り替える。**
③ USB Accessory **モードで接続する。(車載器側が** USB Accessory **機器)**

【注意事項】

① Mirror Link **とは異なり、**USB Accessory **モードでは、スマホ側の画面を車載器に送らないため、車載器は、**USB Accessory **モード時は、自らの画面を表示する必要がある。**
② DRM **の著作権が必要な地デジ等の音声コンテンツは、対応不可となる。**

本書出版時点での市場製品での注意点

4.3章 USB Host APIによる実装において、プラットホームや端末メーカの実装（Android端末のミドル層以上の実装）によっては使用できない、もしくは動作保障されていないAPIがある。