AN986: BLUETOOTH® A2DP AND AVRCP PROFILES

GETTING STARTED GUIDE

Monday, 18 March 2019

Document Revision: 2.2

SILICON LABS

VERSION HISTORY

Date Edited	Comment
1.0	First version
1.1	Added chapter 3.5.4
1.2	Updated to reflect the changes in iWRAP5, corrected errors in chapter 3.1, detailed chapter 3.5.3
1.3	iWRAP5 release updates
1.4	Added AVRCP RAW, iOS behaviour details
1.5	AAC info added to chapter 5, generic iWRAP6 release updates in chapter 2
1.6	iWRAP6 beta release updates: AVRCP 1.5 Controller commands, AVRCP browsing
1.7	iWRAP6 Release Candidate updated: AVRCP 1.5 Target commands (RSP, NFY), added notifications and absolute volume subsection with examples. Restructured most topics and subtopics.
1.8	Update AVRCP PDU events, examples and some minor changes.
1.9	Added information about possibility to enable AVRCP Target and Controller role in the same time
2.0	Added aptX Low Latency feature description
2.1	Updated AVRCP RSP command and related target events
2.2	Updated references

TABLE OF CONTENTS

1 Introduction

This application note discusses *Bluetooth* Advanced Audio Distribution Profile (A2DP) and *Bluetooth* Audio/video Remote Control Profile (AVRCP) their advantages and how these profiles can be used. Also practical examples are given how the A2DP and AVRCP profiles are used with the iWRAP firmware.

1.1 Advanced Audio Distribution Profile

A2DP describes how stereo-quality audio can be streamed from a media source to a sink. The audio source is the music player and the audio sink is the wireless headset or wireless stereo speakers.

The profile defines two roles of an audio device: source and sink.

- **A2DP Source** – A device is the source when it acts as a source of a digital audio stream that is delivered to the SINK of the piconet.
- **A2DP Sink** – A device is the sink when it acts as a sink of a digital audio stream delivered from the SOURCE on the same piconet.

A2DP defines the protocols and procedures that realize distribution of audio content of high-quality in mono or stereo on ACL channels. The term “advanced audio,” therefore, should be distinguished from “*Bluetooth* audio,” which indicates distribution of narrow band voice on SCO channels as defined in the baseband specification.

A2DP profile includes mandatory support for low complexity sub-band codec (SBC) and supports optionally MPEG-1,2 Audio, MPEG-2,4 AAC, ATRAC or other codecs.

The audio data is compressed in a proper format for efficient use of the limited bandwidth. Surround sound distribution is not included in the scope of this profile.

Figure 1: Typical A2DP use case

Source: [1]

1.2 Audio/video Remote Control Profile

AVRCP is designed to provide a standard interface to control TVs, Hi-Fi equipment, or others to allow a single remote controller (or other device) to control all the A/V equipment to which a user has access. It may be used in concert with A2DP or VDP.

Basically your action manipulates the control. You can adjust menu functions that are already commonly used, such as adjusting the brightness of your TV or hue, or a VCR timer, as well as audio functions like sound adjustments, play, pause, skip, etc.

The AVRCP defines two roles, that of a controller and target device.

- **Controller** – The controller is typically considered the remote control device.
- **Target** – The target device is the one whose characteristics are being altered.

In a "walkman" type media player scenario, the control device may be a headset that allows tracks to be skipped and the target device would be the actual medial player.

In AVRCP, the controller translates the detected user action to the A/V control signal, and then transmits it to a remote *Bluetooth* enabled device. The functions available in a conventional infrared remote controller can be realized in this profile. AVRCP Browsing enables browsing media content on a media library –aware player.

Figure 2: Typical AVRCP use case

Source: [2]

2 iWRAP firmware overview

iWRAP is an embedded firmware running entirely on the RISC processor of WT11i, WT12, WT32, WT32i and WT41 modules. It implements the full *Bluetooth* protocol stack and many *Bluetooth* profiles. All software layers, including application software, run on the internal RISC processor in a protected user software execution environment known as a Virtual Machine (VM).

The host system can interface to iWRAP firmware through one or more physical interfaces, which are also shown in the figure below. The most common interfacing is done through the UART interface by using the ASCII commands that iWRAP firmware supports. With these ASCII commands, the host can access *Bluetooth* functionality without paying any attention to the complexity, which lies in the *Bluetooth* protocol stack. GPIO interface can be used for event monitoring and command execution. PCM, SPDIF, I2S or analog interfaces are available for audio. The available interfaces depend on the used hardware.

The user can write application code to the host processor to control iWRAP firmware using ASCII commands or GPIO events. In this way, it is easy to develop *Bluetooth* enabled applications.

On WT32 and WT32i, there is an extra DSP processor available for data/audio processing.

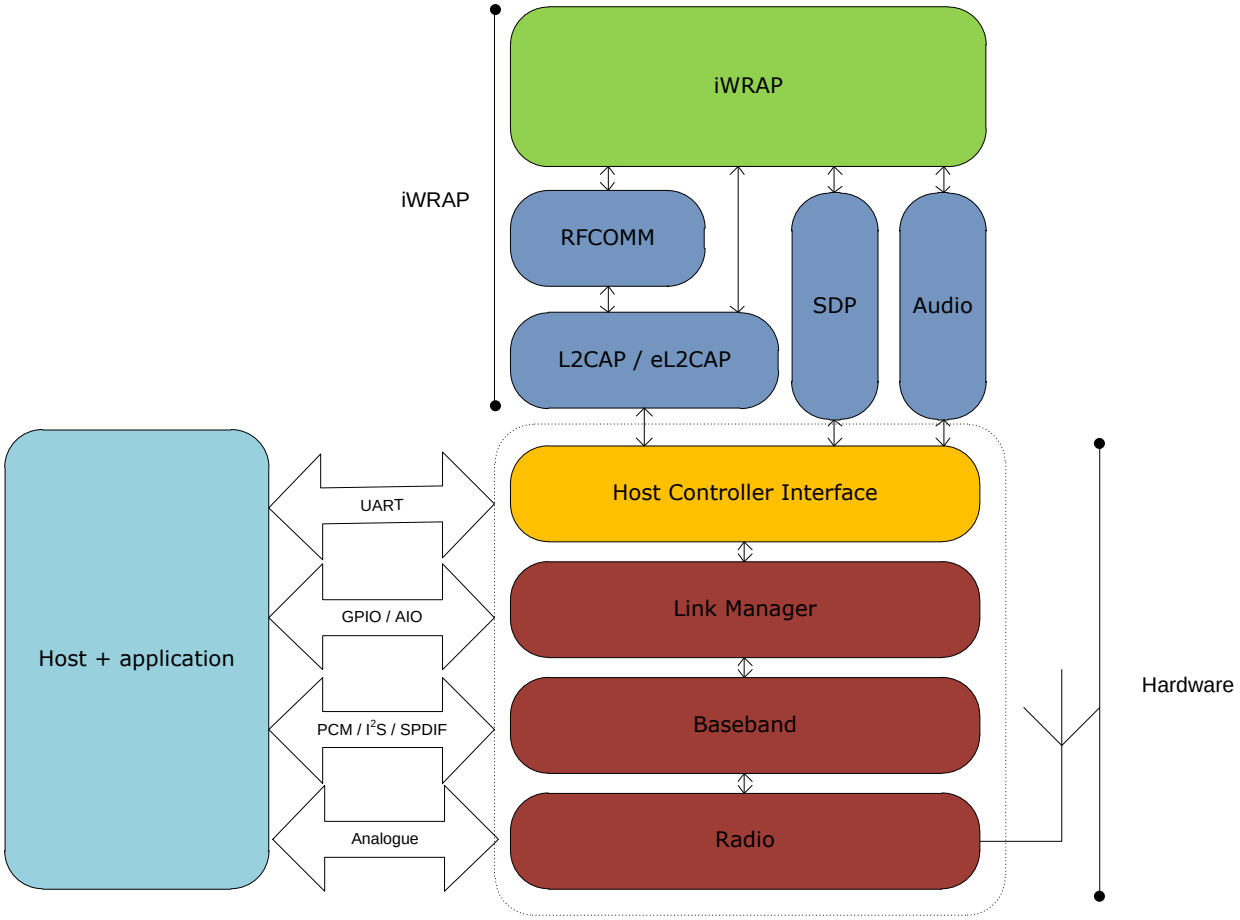


Figure 3: iWRAP *Bluetooth* Stack

In the figure above, a *Bluetooth* module with iWRAP firmware could be connected to a host system for example through the UART interface. The options are:

- If the host system has a processor, software can be used to control iWRAP by using ASCII based commands or GPIO events.
- If there is no need to control iWRAP, or the host system does not need a processor, iWRAP can be configured to be totally transparent and autonomous, in which case it only accepts connections or automatically opens them.
- GPIO lines that *Bluetooth* modules offer can also be used together with iWRAP to achieve additional functionality, such as Carrier Detect or DTR signaling.
- Audio interfaces can be used to transmit audio over a *Bluetooth* link.

3 Using A2DP and AVRCP with iWRAP

This chapter instructs the A2DP and AVRCP usage and configuration with the iWRAP firmware.

3.1 Configuring A2DP

The A2DP profile has only one configuration: whether it is a Sink or a Source. The Sink receives and decodes audio and renders it through one of the audio interfaces of the module. The Source encodes audio, usually received from the ADC or a digital audio interface, and sends it over the Bluetooth link.

To configure iWRAP as the Sink:

SET PROFILE A2DP SINK

To configure iWRAP as the Source:

SET PROFILE A2DP SOURCE

3.2 Configuring AVRCP

The AVRCP profile has two distinct roles, Controller and Target, as described in the first chapter. The Controller always initiates all commands, the Target responds. The Target can send a notification event to the Controller when some variable changes on the Target, but only for events the Controller has previously registered to receive.

In addition to these two roles, AVRCP defines four categories of A/V control commands. Support in the Controller role means support for sending the commands; in the Target role it means support for receiving them.

In iWRAP6, the SET PROFILE AVRCP setting has an additional field (compared with earlier versions of iWRAP): supported categories. The four distinct categories of devices, with their respective mandatory AVRCP commands:

- Category 1: Player/Recorder - must support PLAY and PAUSE
- Category 2: Monitor/Amplifier - must support VOLUP and VOLDN (volume up and down)
- Category 3: Tuner - must support CHUP and CHDN (channel up and down)
- Category 4: Menu - must support MSELECT, MUP, MDN, MLEFT, MRIGHT (menu select and directions)

The syntax is:

SET PROFILE AVRCP {CONTROLLER | TARGET} {features} [{CONTROLLER | TARGET} {features}]

Bits 0-3 in the *features* bit field denote support for Categories 1-4, respectively. At least one Category must be supported.

Bit 4 enable browsing option which is available currently only for controller role. The rest of the bits are reserved for future additions.

Note:

If **{features}** filed is empty, all possible features are being enabled.

In some cases it's also possible that both devices function as Controllers and Targets. iWRAP can broadcasts more than one AVRCP role instance in its SDP record, e.g. it can be Target and Controller in the same time. It is also possible that iWRAP can send Controller commands successfully even when configured as the Target.

For example, if iWRAP is configured as an A2DP Sink + AVRCP Target which supports Absolute Volume control, and an iOS device acts as the A2DP Source, the iOS device will control iWRAP's volume with the media player's volume slider, and will still accept AVRCP commands such as Play, Pause and Stop from iWRAP. In this scenario iWRAP can be thought of as an amplifier or headset capable of locally adjusting output gain, but which also has a panel or buttons for sending Play, Pause etc. commands. The iOS device will adjust its volume level to maximum (that is, no digital attenuation), and let the A2DP Sink adjust the gain locally. This achieves optimal sound quality, as digital gain slightly degrades the audio signal. In case when iWrap device has possibility to control volume level and playback status (Play, Pause, etc.) it is recommended to configure both AVRCP Target and Controller role.

3.3 Example configurations

3.3.1 A2DP Sink and AVRCP Controller

Perhaps the most common use case for A2DP and AVRCP is the wireless headset with simple remote control capabilities: playing, pausing and volume control. The A2DP Sink enables audio streaming from an audio source, whereas AVRCP Controller facilitates over-the-air control of the audio stream.

A2DP Sink is enabled with iWRAP command:

SET PROFILE A2DP SINK

AVRCP controller with support for controlling a player/amplifier application is enabled with command:

SET PROFILE AVRCP CONTROLLER 3

In order that mobile phones, tablets and PCs recognize the device as a wireless headset the *Bluetooth* Class of Device (CoD) must be configured to reflect the device's capabilities. Some devices may not be able to discover and connect the A2DP unless the CoD is properly set.

The A2DP specification mandates that for the A2DP Sink, the Rendering Service bit (0x040000) is set. It is also recommended to set the Audio Service bit (0x200000) and to set the Major Device Class to Audio/Video (0x000400).

The Minor Device Class should be set according to what type of device is in question, for example Headphones (0x000018) and Hi-Fi Audio Device (0x000028). The combined CoD is 0x240438.

Class of device is configured with the iWRAP command:

SET BT CLASS 240438

Finally a reset is needed to for the profiles to become active.

```
SET PROFILE A2DP SINK
SET PROFILE AVRCP CONTROLLER 3
SET BT CLASS 240438
RESET
```

3.3.2 A2DP Source and AVRCP Target

Another very common scenario is an audio source which can be remote controlled, such as a home stereo system.

Much like in the previous example, "**SET PROFILE A2DP SOURCE**" and "**SET PROFILE AVRCP TARGET 7**" must be issued to enable the profiles. In this case, Categories 1 through 3 are supported; a radio tuner is also included in the stereo system.

As in the previous example, the Class of Device must be set properly. In the case of A2DP Source, the specification mandates the use of the Capturing Service bit (0x080000). The Audio Service (0x200000) and Audio/Video Major Device Class (0x000400) should be used, but they are not mandatory.

A common example Minor Device Class is Hi-Fi Audio Device (0x000028). The combined CoD is then 0x280428.

Finally a reset is needed to for the A2DP profile to become active.

```
SET PROFILE A2DP SOURCE
SET PROFILE AVRCP TARGET 7
SET BT CLASS 280428
RESET
```

3.3.3 A2DP Sink and AVRCP Target

In this case, iWRAP is a device that outputs audio, has a volume control and is capable of adjusting audio gain locally, but has no panel with play, pause or other control buttons. An example is a speaker system that receives audio from a phone, and the phone has full control over the audio stream.

```
SET PROFILE A2DP SINK
SET PROFILE AVRCP TARGET 2
SET BT CLASS 240428
RESET
```

3.3.4 A2DP Sink and AVRCP Target and Controller

In this case, iWRAP is a device that outputs audio, has a volume control and is capable of adjusting audio gain locally, but it also has a panel with play, pause or other control buttons. An example is a speaker system that receives audio from a phone, and the both devices has full control over the audio stream.

```
SET PROFILE A2DP SINK
SET PROFILE AVRCP CONTROLLER 3 TARGET 2
SET BT CLASS 240428
RESET
```

3.3.5 Security configuration

To be able to pair with other *Bluetooth* enabled devices the *Bluetooth* security needs to be properly configured. iWRAP support Secure Simple Pairing (SSP) defined in *Bluetooth* 2.1 + EDR specification, and the use of it is mandatory, but PIN code pairing is also supported to enable pairing with legacy devices.

In order to enable SSP and PIN code pairing, the following configuration commands are needed:

"SET BT SSP 3 0" This enables SSP "just works" mode, where no PIN code entry or passkey verification is needed. To enable the optional man-in-the-middle protection, please refer to iWRAP user guide.

"SET BT AUTH * <pin>" This command enables PIN code pairing, to support pairing with legacy defines. **<pin>** is the desired PIN code**,** which can be 1-16 alphanumeric characters.

Finally a reset is needed to for the security settings profile to become active.

Below is an example how to enable SSP just works and PIN code pairing in iWRAP.

```
SET BT SSP 3 0
SET BT AUTH * 0000
RESET
```

3.4 Service discovery

Bluetooth technology enables wireless service discovery, so you can find out the capabilities the remote device supports. Wireless service discovery uses the *Bluetooth* Service Discovery Profile (SDP).

With iWRAP the service discovery is performed with command: "**SDP {*bd_addr*} {*uuid*}**".

bd_addr	*Bluetooth* device address of the remote device.
uuid	Universally unique identifier. Refers to the *Bluetooth* profile one wants to discover. For A2DP sink the ***uuid*** is 110B, for A2DP source 110A for AVRCP target 110C and for AVRCP controller 110E.

Below is an example how to perform a service discovery to an A2DP sink.

SDP 00:07:80:81:66:6f 110B

SDP 00:07:80:81:66:6f < I SERVICENAME S "**Stereo Headset**" > < I PROTOCOLDESCRIPTORLIST < < U L2CAP I **19** > < U 0019 I 100 > > >

SDP

Stereo Headset = Service name

19 = L2CAP psm for A2DP sink

Below is an example how to perform a service discovery to a A2DP source.

SDP 00:07:80:93:0c:aa 110A

SDP 00:07:80:93:0c:aa < I SERVICENAME S "Stereo Audio" > < I PROTOCOLDESCRIPTORLIST < < U L2CAP I 19 > < U 0019 I 100 > > >

SDP

Below is an example how to perform a service discovery to a AVRCP controller.

SDP 00:07:80:93:0c:aa 110C

SDP 00:07:80:81:66:6f < I SERVICENAME S "**A/V Controller**" > < I PROTOCOLDESCRIPTORLIST < < U L2CAP I **17** > < U 0017 I 104 > > >

SDP

A/V Controller = Service name

17 = L2CAP psm for A2DP source

Below is an example how to perform a service discovery to a AVRCP target..

SDP 00:07:80:93:0c:aa 110E

SDP 00:07:80:93:0c:aa < I SERVICENAME S "A/V Target" > < I PROTOCOLDESCRIPTORLIST < < U L2CAP I 17 > < U 0017 I 103 > > >

SDP

3.5 Connection establishment

3.5.1 A2DP connection

A2DP profile requires two connections, one for a control channel and second for a data channel. However iWRAP automatically sets up both the connections so the user does not need to worry about setting up two channels.

The A2DP connection is opened, typical to iWRAP, with a **CALL** command:

"**CALL {*bd_addr*} 19 A2DP**"

bd_addr *Bluetooth* device address of the remote device.

Below is an example how to set up an A2DP connection.

```
CALL 00:07:80:81:66:6f 19 A2DP
CALL 0
CONNECT 0 A2DP 19
CONNECT 1 A2DP 19
A2DP STREAMING START 0
```

A typical indications of outgoing call and successful connection are received (**CALL** and **CONNECT**). Two connection events are typically received indicating the establishment of control and data channels.

Also typically a third event is received indicating that A2DP streaming is started:

"A2DP STREAMING START {*link_id*}"

link_id Numeric connection identifier which started streaming.

Note:

iWRAP allows you to set up several simultaneous A2DP connections. However, only one of the connections can be streaming audio at a time.

3.5.2 AVRCP connection

The AVRCP connection is opened, typical to iWRAP, with a **CALL** command:

"**CALL {*bd_addr*} 17 AVRCP**"

bd_addr *Bluetooth* device address of the remote device.

Below is an example how to set up an AVRCP connection.

```
CALL 00:07:80:81:66:6f 17 AVRCP
CALL 2
CONNECT 2 AVRCP 17
```

Typical indications for outgoing call and successful connection are received (**CALL** and **CONNECT**) and in the case of when the connection establishment is not successful **CALL** and **NO CARRIER** events are received.

Note that since the Controller supports browsing, some devices may automatically connect the browsing channel even if iWRAP does not request it. If browsing functionality is not needed, the browsing channel can be ignored or closed.

3.6 Connection termination

3.6.1 A2DP connection

The A2DP connection is simply closed with iWRAP command "**CLOSE {*link_id*}**". One should always terminate the first A2DP connection (control channel) and not second connection (data channel). Terminating the control channel makes iWRAP also terminate the data channel.

Below is an example of A2DP connection termination.

```
CLOSE 0
A2DP STREAMING STOP 0
NO CARRIER 0 ERROR 0
NO CARRIER 1 ERROR 0
```

Typically first an indication of A2DP streaming stop is received:

"A2DP STREAMING STOP {*link_id*}"

link_id	Numeric connection identifier of link which started streaming.

Then normal **NO CARRIER** events are received for both A2DP connections.

3.6.2 AVRCP connection

The AVRCP connection is simply closed with iWRAP command "**CLOSE {*link_id*}**". If there is a browsing connection open, iWRAP will automatically close the browsing connection first, and then the main AVRCP connection.

Below is an example of AVRCP connection termination.

```
CLOSE 2
NO CARRIER 2 ERROR 0
```

Then normal **NO CARRIER** event is received.

4 AVRCP notifications and absolute volume control

As of iWRAP5, the Controller supports notifications. As of iWRAP6, both the Controller and Target support notifications and absolute volume control.

4.1 Notifications general concepts

In AVRCP, the Controller always initiates all transactions, and the Target responds. For events that happen due to internal events on the Target, or due to user interface, the Target can send notifications. The Controller must initially request for a notification, and when due to an internal event or user interaction something changes on the Target, it will send a notification to the Controller.

The Controller must first ask the Target what kinds of events it supports. Then it may register to receive notifications of one or more of the supported events.

4.2 Absolute volume control general concepts

In AVRCP specification version 1.4, the concept of absolute volume was introduced. Instead of just relative volume commands: volume up and volume down, which cannot take into consideration whether the Target is already at its maximum or minimum volume level, it is also now possible to set and read the volume level of the Target device in percentage. Notifications and absolute volume control work hand in hand, as demonstrated in the next subsection.

4.3 Notifications and absolute volume control workflow

For absolute volume control, it is assumed in the specification that the A2DP Sink is the AVRCP Target. Between two iWRAPs. one configured as Sink+Target and the other as Source+Controller, the exchange might look something like this:

Device A, Source + Controller, connects and sends Get Capabilities PDU 0x10, asking for supported events 0x03

CALL 00:07:80:bb:bb:bb 17 AVRCP

CALL 0

CONNECT 0 AVRCP 17

CALL 00:07:80:bb:bb:bb 19 A2DP

CALL 1

CONNECT 1 A2DP 19

CONNECT 2 A2DP 19

A2DP STREAMING START 1

@0 AVRCP PDU 10 3

Device B, Sink + Target receives the connections and the Get Capabilities PDU. It responds with three events: track changed 0x01, playback status changed 0x02, and (absolute) volume changed 0x0d:

RING 0 00:07:80:aa:aa:aa 17 AVRCP

RING 1 00:07:80:aa:aa:aa 19 A2DP

RING 2 00:07:80:aa:aa:aa 19 A2DP

A2DP STREAMING START 1

AVRCP 0 PDU_GET_CAPABILITIES 3

@0 AVRCP RSP 1 2 d

Device A receives the response for Get Capabilities. Seeing that the other device supports them, it registers for volume changed notifications:

AVRCP 0 PDU_GET_CAPABILITIES_RSP EVENT COUNT 3 TRACK_CHANGED PLAYBACK_STATUS_CHANGED VOLUME_CHANGED

@0 AVRCP PDU 31 d

Device B must now immediately respond with an interim response, which tells the Controller the current value of the volume on the Target, which is 55% of maximum:

AVRCP 0 PDU_REGISTER_NOTIFICATION 1 VOLUME_CHANGED 0

@0 AVRCP NFY INTERIM 1 d 55

Device A receives the interim response, and now it knows the volume is at 55%. It wants to change the volume to 61%:

AVRCP 0 PDU_REGISTER_NOTIFICATION_RSP INTERIM VOLUME_CHANGED 55

@0 AVRCP PDU 50 61

Device B receives the command, but since it can only change volume in 5% increments, it adjusts its volume to the closest number, 60%, and reports it back to the Controller. Note that this will not trigger a volume changed notification, because the Controller requested for the change.

AVRCP 0 PDU_SET_ABSOLUTE_VOLUME 61

@0 AVRCP RSP 60

Device A now sees the volume was set to 60%, and adjusts its own volume display accordingly.

AVRCP 0 PDU_SET_ABSOLUTE_VOLUME_RSP 60

The user of Device B turns back the volume by one step, bringing it down to 55%. This must be notified to the Controller, and triggers a Changed notification. Note that the transaction label (first parameter after the INTERIM string) is the same as in the original request.

@0 AVRCP NFY CHANGED 1 d 55

Now, if Device A still wants to monitor the volume level on Device B, it must register for another notification, because as per the AVRCP specification all notifications are triggered only once.

5 AVRCP browsing

Browsing is a feature introduced in AVRCP specification version 1.4. iWRAP6 implements AVRCP version 1.5 Controller, so browsing is supported on the Controller side.

AVRCP browsing is used to navigate the virtual file system on any media player on the Target, where browsing is supported. AVRCP browsing facilitates listing and selecting media players, navigating up and down folders and listing folder contents, retrieving information about items on the player, and searching for specific items.

5.1 Browsing general concepts

5.1.1 Browsing channel

Browsing is done over a separate L2CAP channel, which is opened after the regular AVRCP channel has been established. All regular AVRCP commands will still be sent over the regular AVRCP channel. The browsing channel is only used for retrieving information about media on the Target device; all other actions, such as adding found items to the now playing list, will still be executed over the regular AVRCP channel.

Either the Controller or the Target can open the browsing channel, and the channel can be opened and closed on demand, or more commonly, kept open as long as the regular AVRCP link is open. The browsing channel must be closed before the regular AVRCP link is closed.

The browsing channel's connection type in the iWRAP LIST output is AVRCP, but its L2CAP PSM number is 0x001b, whereas the PSM of the regular AVRCP connection is 0x0017.

5.1.2 UIDs and UID counter

Every item on a browsing-capable player has a unique 128-bit UID that is valid over both AVRCP channels. Any item discovered over the browsing channel can be addressed with regular AVRCP PDUs (for example, PDU 0x74, PLAY_ITEM) by using the discovered UID.

A media item's UID may change if more media items are added to or removed from the player's virtual filesystem. The UID counter is a mechanism to keep track of UID changes. Every time the Controller sends a AVRCP BROWSE LIST or AVRCP BROWSE SEARCH browsing command, the Target returns, along with other data, the current value of the UID counter. Every Controller command that addresses media items by UID must also include the last UID counter value the Controller has received.

If the UID counter values do not match (that is, something has been added, removed, or changed on the player's virtual filesystem), the Target will reject the command with status 0x05, UIDs changed. The Controller should then repeat the LIST command it used to retrieve the UID of the item.

5.2 Browsing workflow

Since a Target can have multiple media players that support browsing, the first step is to get a list of media players and select a player for browsing.

Example: open AVRCP and AVRCP browsing links, and request the first 16 items on the media player list, then set the player with ID **0x0001** as the browsed player.

CALL 00:ff:00:bb:aa:cc 17 AVRCP

CALL 0

CONNECT 0 AVRCP 17

@0 AVRCP BROWSE OPEN

CONNECT 1 AVRCP 1b

@0 AVRCP BROWSE LIST 0 0 f ff

AVRCP_BROWSE 1 GET_FOLDER_ITEMS UIDC 0000 ITEMS 0001 < PLAYER ID **0001** TYPE 01 SUBTYPE 00000000 PLAY_STATUS STOPPED FEATURES 0000000000b7011c0200000000000000 NAME 0005 Music >

@0 AVRCP BROWSE SETPLAYER 1

AVRCP_BROWSE 1 SET_BROWSED_PLAYER UIDC 0000 COUNT 00000005 FOLDERS 00

Note:

If the AVRCP browse channel is already open (for example opened up by the phone) the **AVRCP BROWSE OPEN** will return syntax error.

After a player is selected for browsing, it's time to retrieve the folder listing and start looking for media to play.

@0 AVRCP BROWSE LIST 1 0 f ff

AVRCP_BROWSE 1 GET_FOLDER_ITEMS UIDC 0000 ITEMS 0005 < FOLDER < UID 0400000000000000 TYPE ARTISTS PLAYABLE NO NAME 0007 Artists > FOLDER < UID 0500000000000000 TYPE TITLES PLAYABLE NO NAME 0005 Songs > FOLDER < UID 0600000000000000 TYPE ALBUMS PLAYABLE NO NAME 0006 Albums > FOLDER < UID 0900000000000000 TYPE ARTISTS PLAYABLE NO NAME 0009 Composers > FOLDER < UID 0a00000000000000 TYPE GENRES PLAYABLE NO NAME 0006 Genres > >

Suppose we want to navigate directly to the songs folder and select some items to put in the playing queue. The songs folder contains 1509 items.

@0 AVRCP BROWSE PATH 0 1 0500000000000000

AVRCP_BROWSE 1 CHANGE_PATH COUNT 000005e5

Next, let's list 3 media items starting from the 2nd item, and ask for all attributes.

@0 AVRCP BROWSE LIST 1 1 3 ff

AVRCP_BROWSE 3 GET_FOLDER_ITEMS UIDC 0000 ITEMS 0003 < < MEDIA UID 9b1afa9f4c12ce59 TYPE AUDIO NAME 0003 ABC ATTRIBUTES 01 < < TITLE NAME 0003 ABC > > > < MEDIA UID 2bf8243352cebe67 TYPE AUDIO NAME 000a About Life ATTRIBUTES 01 < < TITLE NAME 000a About Life > > > < MEDIA UID b0bdea9e7ecb5224 TYPE AUDIO NAME 0008 Absorbed ATTRIBUTES 01 < < TITLE NAME 0008 Absorbed > > > >

Now we'll start playing the song titled "ABC".

@0 AVRCP PDU 74 1 9b1afa9f4c12ce59 0

AVRCP 0 PLAY_ITEM_RSP OK

Let's search for an album called "Rain Dogs" to put in the playback queue, and then retrieve the 10 first search results.

@0 AVRCP BROWSE SEARCH "rain dogs"

AVRCP_BROWSE 1 SEARCH UIDC 0000 COUNT 0000001b

@0 AVRCP BROWSE LIST 3 0 9 0

AVRCP_BROWSE 1 GET_FOLDER_ITEMS UIDC 0000 ITEMS 000a < < MEDIA UID 0e24d548196f8e70 TYPE AUDIO NAME 0016 Anywhere I Lay My Head ATTRIBUTES 01 < < TITLE NAME 0016 Anywhere I Lay My Head > > > < MEDIA UID 8942c6d007e09367 TYPE AUDIO NAME 0010 Big Black Mariah ATTRIBUTES 01 < < TITLE NAME 0010 Big Black Mariah > > > < MEDIA UID 884472822381fb9b TYPE AUDIO NAME 000a Blind Love ATTRIBUTES 01 < < TITLE NAME 000a Blind Love > > > < MEDIA UID 06eec7217eed9522 TYPE AUDIO NAME 0011 Bride of Rain Dog ATTRIBUTES 01 < < TITLE NAME 0011 Bride of Rain Dog > > > < MEDIA UID bab8b9091baf783a TYPE AUDIO NAME 000e Cemetery Polka ATTRIBUTES 01 < < TITLE NAME 000e Cemetery Polka > > > < MEDIA UID 04bb90d8c8c54d00 TYPE AUDIO NAME 000a Clap Hands ATTRIBUTES 01 < < TITLE NAME 000a Clap Hands > > > < MEDIA UID 163f19ccd93ee837 TYPE AUDIO NAME 000f Diamonds & Gold ATTRIBUTES 01 < < TITLE NAME 000f Diamonds & Gold > > > < MEDIA UID f73b53fa79c152b0 TYPE AUDIO NAME 0016 Do You Close Your Eyes ATTRIBUTES 01 < < TITLE NAME 0016 Do You Close Your Eyes > > > < MEDIA UID 97526ba94b21557c TYPE AUDIO NAME 000e Downtown Train ATTRIBUTES 01 < < TITLE NAME 000e Downtown Train > > > < MEDIA UID db0b9bfc5abaa800 TYPE AUDIO NAME 000f Gun Street Girl ATTRIBUTES 01 < < TITLE NAME 000f Gun Street Girl > > > >

Let's add the first song in the search results to the playback queue.

@0 AVRCP PDU 90 2 0e24d548196f8e70 0

AVRCP 0 ADD_TO_NOW_PLAYING_RSP OK

6 A2DP and AVRCP command reference

This chapter contains general information and tips about the iWRAP and A2DP profile for the implementers.

6.1 A2DP commands

The table below lists the possible A2DP commands. See iWRAP user guide for exact details.

Command	Function
A2DP STREAMING START	Starts A2DP streaming
A2DP ATREAMING STOP	Stops A2DP streaming

Table 1: Supported A2DP commands

6.2 AVRCP passthrough commands

The table below lists the possible AVRCP passthrough commands.

Note:

Not all devices support all commands. For example, iOS devices do not support remote volume control using **AVRCP UP**, or **AVRCP DN**, or **AVRCP MUTE** commands. This is a limitation of iOS and not an iWRAP firmware problem.

Command	Description
AVRCP UP	Volume up
AVRCP DN	Volume down
AVRCP MUTE	Mute
AVRCP STOP	Stop
AVRCP PLAY	Play
AVRCP PAUSE	Pause
AVRCP REWIND	Rewind
AVRCP FAST_FORWARD	Fast Forward
AVRCP FORWARD	Forward (next song)
AVRCP BACKWARD	Backward (previous song)
AVRCP {*raw*}	AVRCP command in raw hex mode. (See section 6.7) **Available codes:** 00-13, 20-2c, 30-37, 40-4b, 50-51, 71-75 and 7e
AVRCP PDU {*PDU_ID*} [*parameters*]	***PDU_ID*** AVRCP PDU ID ***parameters*** AVRCP PDU parameters Please refer to the following page for available ***PDU_IDs*** and ***parameters.***

Table 2: Available AVRCP commands

Note: See iWRAP user guide for more details and examples of how to use the AVRCP commands.

6.3 AVRCP PDU

6.3.1 Syntax

AVRCP PDU command is used by the AVRCP Controller to send metadata request Protocol Data Units to the Target.

Synopsis
AVRCP PDU {*PDU_ID*} [*parameters*]

PDU ID	Description and parameters
10	Get capabilities command. Query for events or Company_ID's the Target supports. **Parameters**: **2** Query supported Company_ID's. **3** Query supported events.
11	List player application settings. No parameters.
12	List possible values for a player application setting. **Parameters**: **{setting_id}** See list at the end of this command's description.
13	Get current values of player application settings. **Parameters**: **{number of settings}** Number of following parameters. Followed by: **{setting_id}** See list at the end of this command's description.
14	Set current values of player application settings. **Parameters**: **{number of settings}**

	Number of setting_id-value-pairs that follow. Followed by: **{setting_id} {value}** See list at the end of this command's description.
20	Get attributes of the currently playing track. **Parameters**: **{number of attributes}** Number of attributes that follow. If zero, list all available information. Followed by (unless number of attributes is zero): **[attribute_id]** See list at the end of this command's description.
30	Get the playing status, length and position of the current track. No parameters.
31	Register notification of events. This will request the Target to notify us when a track is changed for instance. **Parameters**: **{event_id}** See list at the end of this command's description.
50	Set the absolute volume on the target device. **Parameters**: **{volume}** The requested volume level in percentage, range 0-100.
60	Set the addressed player. If the Target has multiple media players in the media player list, you can select which player should receive the AVRCP commands. **Parameters**: **{Player ID}** The ID of the player
74 ***90***	74: Start playing a track immediately. 90: Add track to the now playing list (playback queue). **Parameters**: **{scope}** 0: Media player list 1: Virtual filesystem 2: Search results

	3: Now playing list **{UID}** UID of the media item. Use the browsing channel to discover UIDs. **{UID counter}** Latest UID counter value received. See section 3.5.1.2 for details.

Note: The AVRCP profile specification contains the details of the possible PDU ID and a detailed description of the parameters.

Note: Not all the AVRCP commands are supported by the AVRCP target devices and most of them actually implement only a small subset of all the available commands.

6.3.2 Events

AVRCP PDU command may return the following events

Events
AVRCP {link_id} {*PDU_ID name*}_RSP [*parsed data*]
AVRCP {link_id} RSP PDU_ID {*PDU_ID*}, data: [*unparsed data*]
AVRCP {link_id} {*PDU_ID name*}_RSP REJ

6.3.3 List of IDs

List of supported **attribute_id**s, **event_id**s, **setting_id**s, and their respective **value**s.

Type	ID	Explanation
Attribute	**0x01**	Song title
Attribute	**0x02**	Artist
Attribute	**0x03**	Album title
Attribute	**0x04**	Track number in numeric ASCII text
Attribute	**0x05**	Total number of tracks on album in numeric ASCII text
Attribute	**0x06**	Genre
Attribute	**0x07**	Playing time in milliseconds in numeric ASCII text
Setting	**0x01**	Equalizer setting, values: **0x01** Off **0x02** On
Setting	**0x02**	Repeat setting, values: **0x01** Off **0x02** Single track **0x03** All tracks **0x04** Group
Setting	**0x03**	Shuffle setting, values: **0x01** Off **0x02** All tracks **0x03** Group
Setting	**0x04**	Scan setting, values: **0x01** Off **0x02** All tracks **0x03** Group
Event	**0x01**	Playback status changed
Event	**0x02**	Track changed

Event	**0x03**	Track reached end
Event	**0x04**	Track reached start
Event	**0x05**	Playback position changed
Event	**0x06**	Battery status changed
Event	**0x07**	System status changed
Event	**0x08**	Player application setting changed
Event	**0x09**	Now playing changed
Event	**0x0a**	Available players changed
Event	**0x0b**	Addressed players changed
Event	**0x0c**	UIDs changed
Event	**0x0d**	Volume changed

6.3.4 Examples

Those examples assume that AVRCP is on link 3.

Query supported AVRCP events.

AVRCP PDU 10 3

AVRCP 3 GET_CAPABILITIES_RSP EVENT COUNT 3 PLAYBACK_STATUS_CHANGED TRACK_CHANGED PLAYBACK_POSITION_CHANGED

Ask the Target about its player application settings, their possible values and change a value.

AVRCP PDU 11

AVRCP 3 LIST_APPLICATION_SETTING_ATTRIBUTES_RSP COUNT 2 REPEAT SHUFFLE

AVRCP PDU 12 2

AVRCP 3 LIST_APPLICATION_SETTING_VALUES_RSP COUNT 3 1 2 3

AVRCP PDU 13 1 2

AVRCP 3 GET_APPLICATION_SETTING_VALUE_RSP COUNT 1 REPEAT OFF

AVRCP PDU 14 1 2 2

AVRCP 3 SET_APPLICATION_SETTING_VALUE_RSP OK

AVRCP PDU 13 1 2

AVRCP 3 GET_APPLICATION_SETTING_VALUE_RSP COUNT 1 REPEAT SINGLE_TRACK

Ask the Target about the title and artist of the song that is currently playing and ask it to notify us if the playback status changes.

AVRCP PDU 20 2 1 2

AVRCP 3 GET_ELEMENT_ATTRIBUTES_RSP COUNT 2 TITLE 24 “Cold Women and Warm Beer” ARTIST 16 “The Black League”

AVRCP PDU 31 1 1

AVRCP 3 REGISTER_NOTIFICATION_RSP INTERIM PLAYBACK_STATUS_CHANGED PLAYING

(the interim response is received right after the request to confirm we were registered for notification)

AVRCP 3 REGISTER_NOTIFICATION_RSP CHANGED PLAYBACK_STATUS_CHANGED PAUSED

(the changed response is received when the playing status changes)

6.4 AVRCP RSP

6.4.1 Syntax

AVRCP RSP is used by the Target to respond to a received AVRCP Controller PDU command. All PDU commands must be responded to. Passthrough commands (PLAY, PAUSE etc.) will be handled automatically by iWRAP, and require no user interaction.

The required parameters depend on which PDU the AVRCP RSP is responding to.

Synopsis
AVRCP RSP [parameters]

PDU received	Description and parameters
GET_CAPABILITIES 2	The Controller is querying for supported Company IDs **AVRCP RSP [id0 id1 …]** **[id0 id1 …]** List of IrDA Company IDs. Usually, the list can be left empty, as iWRAP will automatically fill in the Bluetooth SIG company ID.
GET_CAPABILITIES 3	The Controller is querying for supported Event IDs **AVRCP RSP [id0 id1 …]** **[id0 id1 …]** List of events supported for notifications. These are listed in 6.3.3 “List of IDs”.
LIST_APPLICATION_SETTING_ATTRIBUTES	The Controller is querying for supported player application setting attributes. **AVRCP RSP [id0 id1 …]** **[id0 id1 …]** List of supported settings’ IDs. These are listed in 6.3.3 “List of IDs”.
LIST_APPLICATION_SETTING_VALUES *{id}*	The Controller is querying for set of possible values for the requested player application setting ID ***{id}***. **AVRCP RSP [val0 val1 …]** **[val0 val1 …]** List of possible values for requested setting ID. These are listed in 6.3.3 “List of IDs”.

GET_APPLICATION_SETTING_VALUE ***{count} [id0 id1 …]?***	The Controller is querying for the current values set on the target for the provided player application setting IDs. ***{count}*** is a number of IDs Target is queried about. **AVRCP RSP [val0 val1 …]** **[val0 val1 …]** List of currently set player application values on the target for the corresponding requested player application setting IDs. These values shall be from set of possible values returned in response to LIST_APPLICATION_SETTING_VALUES command.
SET_APPLICATION_SETTING_VALUE ***{count} [id0 val0 id1 val1 …]?***	The Controller requests to set the list of player application setting values (***val0 val1 …***) on the target device for the corresponding defined list of player application setting IDs (***id0 id1 …***). If setting IDs and values are correct, target sends the accept answer. ***{count}*** is a number of IDs Target is requested to set values of. Otherwise **SET_APPLICATION_SETTING_VALUE** event with **ILLEGAL** or **UNKNOWN** parameter is received and target sends the reject answer.
GET_ELEMENT_ATTRIBUTES {count} [id0 id 1 …]	Get attributes of the current track. The first response parameter should be the same count of attributes asked in the PDU. The rest of the parameters are a list containing pairs of numeric IDs and quote-enclosed strings, which contain the IDs and the the text values of the attributes asked. The attribute IDs are listed in 6.3.3 “List of IDs”. **AVRCP RSP {count} [id0 “value0” id1 “value1” …]** **{count}** Number of following attributes. **[id0 “value0” id1 “value1” …]** ID-text pairs of attributes. The maximum length for any single attribute value is 255 bytes (excluding the enclosing quotes).
GET_PLAY_STATUS	The Controller is querying for the current playing status and track playback position. **AVRCP RSP {length} {position} {status}** **{length}** Length of the current track in milliseconds, hexadecimal format. If no track is selected or song length reporting is not supported, return 0xffffffff. **{position}** Playback position of the current track in milliseconds, hexadecimal format. If no track is selected or playback position reporting is not supported, return 0xffffffff.

	{status} Playback status: **0** - Stopped **1** - Playing **2** - Paused **3** - Fast forwarding **4** - Rewinding
SET_ABSOLUTE_VOLUME {volume}	The Controller set the absolute volume level. The exact volume level which was set must be reported back, due to the Target possibly having different volume adjustment granularity than the Controller. **AVRCP RSP {volume}** **{volume}** The volume that was set on the Target.

6.4.2 Examples

Controller asks for three 3 attributes of the current song: 1 (the title), 2 (artist) and 3 (album). Target responses with track details:

AVRCP 0 PDU_GET_ELEMENT_ATTRIBUTES 3 1 2 3

AVRCP RSP 3 1 “Some Title” 2 “Some Artist” 3 “Some Album”

Controller asks for the supported settings. Target responses that settings **1** (Equalizer) and **2** (Repeat) are supported:

AVRCP 0 LIST_APPLICATION_SETTING_ATTRIBUTES

AVRCP RSP 1 2

Controller asks for the supported values of Equalizer setting (0x01). Target responses that available values for Equalizer setting are **1** (Off) and **2** (On) values after:

AVRCP 0 LIST_APPLICATION_SETTING_VALUES 1

AVRCP RSP 1 2

Controller asks for current values of 0x02 settings: Equalizer (0x01) and Repeat (0x02). Target responses that Equalizer is set to **1** (Off) and Repeat is set to **2** (Single track):

AVRCP 0 GET_APPLICATION_SETTING_VALUE 02 01 02?

AVRCP RSP 1 2

Controller sets on Target 0x02 settings: 0x01 (Equalizer) to 0x01 (Off) and 0x02 (Repeat) to 0x01 (Off):

AVRCP 0 SET_APPLICATION_SETTING_VALUE 02 01 01 02 01

6.5 AVRCP NFY

6.5.1 Syntax

AVRCP NFY is used by the Target to respond to a received notification request. Each notification request must be responded to immediately with an AVRCP NFY INTERIM command, and when the notification is triggered, it must be reported with AVRCP NFY CHANGED.

Each AVRCP command-response pair carries its own transaction label to help the Controller differentiate between responses received from the Target. ***The transaction label in both the NFY INTERIM and NFY CHANGED responses must match that of the original notification request.*** With other commands, iWRAP will automatically handle the updating of the transaction label, but in the notification commands, the user must supply it, because iWRAP cannot know when an event that triggers a NFY CHANGED will happen.

Notification request event format:

AVRCP {link_id} PDU_REGISTER_NOTIFICATION {transaction_label} {event} {parameter}

Currently, the parameter is not used in any supported notification, but may be used in the future.

Synopsis
AVRCP NFY {INTERIM | CHANGED} {transaction_label} {event_ID} [value]

Event registered	Description and parameters
PLAYBACK_STATUS_CHANGED 0	**{event_ID}** 1 **[value]** Status of playback, see list under GET_PLAY_STATUS in 6.3.1 “AVRCP PDU Syntax”
TRACK_CHANGED 0	**{event_ID}** 2 **[value]** 0 – no track is currently selected 1 – a track is currently selected
VOLUME_CHANGED 0	**{event_ID}** d **[value]**

	Current volume level on Target, percentage in decimal.

6.5.2 Examples

Respond with playback status changes.

AVRCP 0 PDU_REGISTER_NOTIFICATION 4 PLAYBACK_STATUS_CHANGED 0 **AVRCP NFY INTERIM 4 1 0** (transaction label 4, playback status changed, status “stopped”) (user presses Play on the Target device, playback status changes to “playing”) **AVRCP NFY CHANGED 4 1 1**

6.6 AVRCP BROWSE

6.6.1 Syntax

AVRCP BROWSE is used to browse an AVRCP Target's virtual filesystem.

Synopsis
AVRCP BROWSE {cmd [params]}

cmd	Description and parameters
OPEN	Open the browsing channel, if it's not already present. The browsing channel can be closed with the **CLOSE** command when necessary, and it will not automatically close the regular AVRCP link. The browsing channel can be used immediately after it has opened. **Parameters**: None
LIST	Lists items in a given scope. The start_item and end_item parameters are used to retrieve the folder contents in chunks, if all items do not fit inside a single packet. **Parameters**: **{scope}** 0: Media player list 1: Virtual filesystem 2: Search results 3: Now playing list **{start_item}** The running number of the first item inside the folder to retrieve. **{end_item}** The running number of the last item to retrieve. **{number_of_attributes}** Number attributes to retrieve for the items. 0: All attributes ff: No attributes **[attribute_list]** Sequence of attributes; omit if number_of_attributes is 0 or ff. The supported values for attribute IDs are listed in the table in chapter 6.3.3

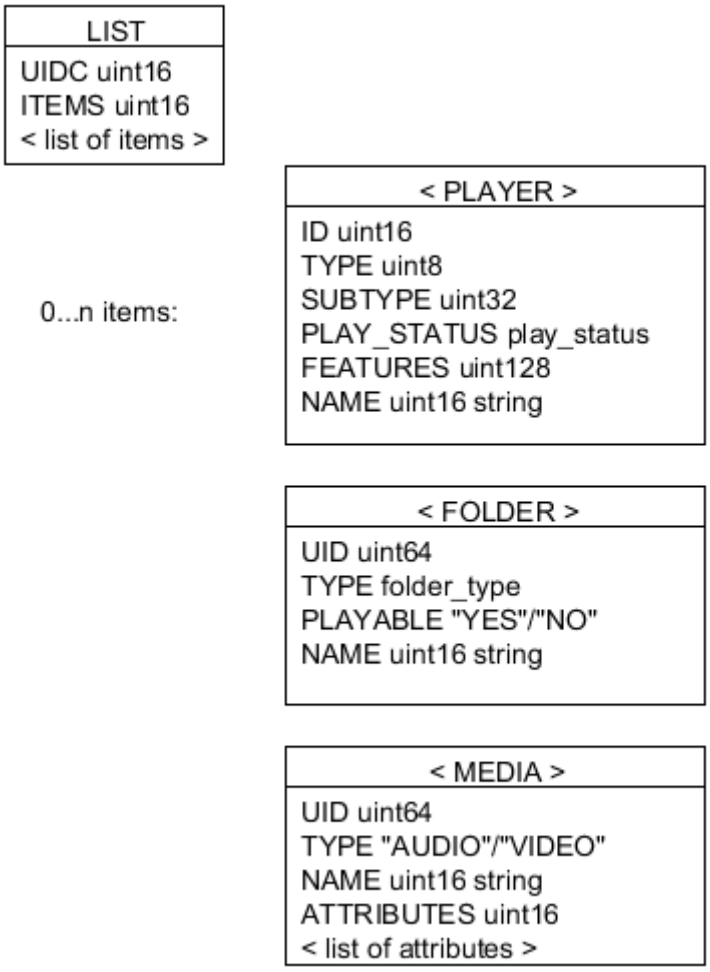
SETPLAYER	Selects the player to browse. A Target may have multiple players, so the player must be selected prior to browsing its virtual filesystem. **Parameters**: **{ID}** 16-bit hexadecimal ID of the player
PATH	Navigate up or down inside the virtual folder structure. **Parameters**: **{UID_counter}** UID counter – see the description in chapter 3.5 **{direction}** 0: up – folder UID is omitted, as it's implicitly known 1: down **[folder_UID]** UID of the target folder (omitted if direction is 0)
GET	Get attributes of an item **Parameters**: **{scope}** 0: Media player list 1: Virtual filesystem 2: Search results 3: Now playing list **{UID_counter}** UID counter – see the description in chapter 5.1 **{UID}** UID of the item **{number_of_attributes}** Number attributes to retrieve for the items. 0: All attributes ff: No attributes **[attribute_list]** Sequence of attributes; omit if number_of_attributes is 0 or ff. The supported values for attribute IDs are listed in the table in chapter 6.3.3
SEARCH	Search for all items containing the search string. No wildcards are accepted. The search results will appear in the search results scope.

	Parameters: **{“search_string”}** String enclosed in quotes.

6.6.2 Response data format

The only response for **AVRCP BROWSE OPEN** is the browsing channel opening or failing to open.

The response for **AVRCP BROWSE LIST** contains the current value of the UID Counter and a list of items, which can be of three distinct types: Player, Folder or Media.

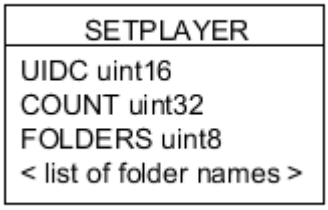


For example, an AVRCP BROWSE LIST which asks for items #4, #5 and #6 in the current virtual filesystem folder, and the title and artist attribute (if applicable), response may look like this:

@1 AVRCP BROWSE LIST 1 4 6 2 1 2

AVRCP_BROWSE 1 GET_FOLDER_ITEMS UIDC 0042 ITEMS 0004 < FOLDER < UID 0000000000000010 TYPE TITLES PLAYABLE YES NAME 0009 “Favorites” > < FOLDER UID 0000000000000020 TYPE ARTISTS PLAYABLE NO NAME 0007 “Artists” > < MEDIA UID 000000000000003f TYPE AUDIO NAME 000c “SomeSong.mp3” ATTRIBUTES 0002 < < TITLE NAME 0008 “SomeSong” > < ARTIST NAME 000a “SomeArtist” > > > >

The response for **AVRCP BROWSE SETPLAYER** returns the current UID Counter, the count of items in the current folder, and if the current folder is not the root folder (for example, a player might start browsing in a folder called /Artists/Albums/ by default), the count and names of the folders leading to the current folder from the root.

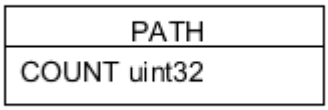


Example: select a player with ID 2 as the browsed player. The player has 32 media items, and it is currently in the folder /Playlists/Favorite/

@1 AVRCP BROWSE SETPLAYER 2

AVRCP_BROWSE 1 SETPLAYER UIDC 0001 COUNT 00000020 FOLDERS 02 < NAME 0009 “Playlists” NAME 0008 “Favorite” >

The response for **AVRCP BROWSE PATH** returns the count of items in the current folder.

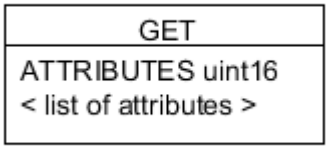


Example: browse down into a folder with UID 0x0000000000000022, when the last received UID Counter value was 0x0042.

@1 AVRCP BROWSE PATH 42 1 22

AVRCP_BROWSE 1 SETPLAYER UIDC 0001 COUNT 00000020 FOLDERS 02 < NAME 0009 “Playlists” NAME 0008 “Favorite” >

The response for **AVRCP BROWSE GET** returns a list of attributes.

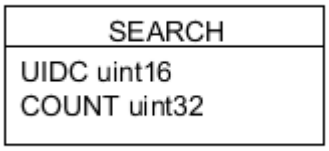


Example: get all attributes for an item with UID 0x33 in the virtual filesystem, last received UID Counter value 0x0042.

@1 AVRCP BROWSE GET 1 42 33 0

AVRCP_BROWSE 1 GET ATTRIBUTES 0003 < < TITLE NAME 0004 “Song” > < ARTIST NAME 0006 “Artist” > < ALBUM NAME 0005 “Album” > >

The response for **AVRCP BROWSE SEARCH** returns the current UID Counter value and the number of search results that are now in the Search Results scope.



Example: search for all items containing the string “abcd”; 3 items match. More information about the matching items can be retrieved with AVRCP BROWSE LIST in the search results scope.

@1 AVRCP BROWSE SEARCH “abcd”

AVRCP_BROWSE 1 SEARCH UIDC 0042 COUNT 0003

6.6.3 Errors

Every browsing command with the exception of AVRCP BROWSE OPEN can return an error code, or in the case of AVRCP BROWSE SEARCH, a General Reject, if the Target does not support searching, which is an optional feature.

@1 AVRCP BROWSE SEARCH “abcd”

AVRCP_BROWSE 1 GENERAL_REJECT 0f

If a command fails for some other reason, for instance if the UIDs have changed on the Target, which happens when media files are modified on, or added to or removed from, the Target’s filesystem.

@1 AVRCP BROWSE GET 1 42 33 0

AVRCP_BROWSE 1 GET ERROR 05

The error codes are listed in the following table:

Status code	Explanation
0x00	Invalid command – the Target did not understand the command
0x01	Invalid parameter – the command was recognized, but one or more of the parameters were not
0x02	Parameter content error – a parameter was recognized, but contained invalid content, for example the Controller tried to change a supported setting to an unsupported value
0x03	Internal error – the command was understood, but could not be completed due to some internal condition
0x04	Success – this value will never be reported in an error
0x05	UIDs changed – the UID Counter issued by the Controller did not match Target's UID Counter number. The Controller should refresh its cached UIDs.
0x06	The command ID is reserved
0x07	Invalid direction – applicable only for AVRCP BROWSE PATH
0x08	Not a directory – applicable only for AVRCP BROWSE PATH
0x09	Does not exist – no such media item or folder found with requested UID
0x0a	Invalid scope
0x0b	Out of bounds – a parameter was too large, for example the AVRCP BROWSE LIST end_item parameter was larger than the number of items in the folder
0x0c	Item not playable – the Controller attempted to play a folder that is not playable
0x0d	Media in use – some media cannot be played if they are in use, for example a CD or DVD
0x0e	Now playing list full – cannot add any more items to the now playing list
0x0f	Search not supported – search is an optional feature, and all Targets may not have implemented it
0x10	Search in progress – cannot start another search before the previous search is finished
0x11	Invalid player ID – applicable only for AVRCP BROWSE SETPLAYER
0x12	Player not browseable – while the Target supports browsing, the selected player

	cannot be browsed
0x13	Player not addressed – the Target has multiple players, and does not know which player should receive the command; the Controller should use AVRCP PDU 60 to select the addressed player
0x14	No valid search results
0x15	No available players – no player can complete the request at the moment
0x16	Addressed player changed – the addressed player has changed for some internal reason, such as the user switching off a media player and starting another on the Target

6.7 AVRCP {raw} detail

The "**AVRCP {raw}**" command is used to send raw AVRCP press/release codes instead of using the shortcut commands such as "**AVRCP PLAY**" or "**AVRCP FAST_FORWARD**" to accomplish the same goals. The shortcut commands are, in fact, simply automated press/release combinations of each command code. It is possible to achieve the same functionality with two unique {raw} commands, and this also gives you more control over functions which benefit from more precisely timed "press" and "release" actions, such as rewind and fast-forward.

The list of supported AVRCP command codes and their meanings are in the table below. Remember that **you must send a "release" command to correspond with every "press" command in order to ensure proper functionality.**

Each command code has a 7-bit value, ranging between 0x00 and 0x7F. The 8th bit (0x80) is a mask which is used to indicate whether it is a "press" code (8th bit cleared) or a "release" code (8th bit is set).

Note:

Not all devices support all command code. For example, iOS devices do not support remote volume control using **0x41, 0x42,** or **0x43** commands. This is a limitation of iOS and not an iWRAP firmware problem.

Press code (hex)	Release code (hex)	Description
00	**80**	Select
01	**81**	D-pad up
02	**82**	D-pad down
03	**83**	D-pad left
04	**84**	D-pad right
05	**85**	D-pad up/right
06	**86**	D-pad down/right
07	**87**	D-pad up/left
08	**88**	D-pad down/left
09	**89**	Root menu
0A	**8A**	Setup menu
0B	**8B**	Contents menu
0C	**8C**	Favourites menu
0D	**8D**	Exit
0x0E – 0x1F reserved		
20	**A0**	0 number entry

21	**A1**	1 number entry
22	**A2**	2 number entry
23	**A3**	3 number entry
24	**A4**	4 number entry
25	**A5**	5 number entry
26	**A6**	6 number entry
27	**A7**	7 number entry
28	**A8**	8 number entry
29	**A9**	9 number entry
2A	**AA**	Dot (.) entry
2B	**AB**	Enter
2C	**AC**	Clear
0x2D – 0x2F reserved		
30	**B0**	Channel up
31	**B1**	Channel down
32	**B2**	Previous channel
33	**B3**	Sound select
34	**B4**	Input select
35	**B5**	Display information
36	**B6**	Help
37	**B7**	Page up
38	**B8**	Page down
0x39 – 0x3F reserved		
40	**C0**	Power
41	**C1**	Volume up
42	**C2**	Volume down
43	**C3**	Mute

44	**C4**	Play
45	**C5**	Stop
46	**C6**	Pause
47	**C7**	Record
48	**C8**	Rewind
49	**C9**	Fast-forward
4A	**CA**	Eject
4B	**CB**	Forward (next track)
4C	**CC**	Backward (previous track)
0x4D – 0x4F reserved		
50	**D0**	Angle
51	**D1**	Subtitle
0x52 – 0x70 reserved		
71	**F1**	F1 function key
72	**F2**	F2 function key
73	**F3**	F3 function key
74	**F4**	F4 function key
75	**F5**	F5 function key

Below is an example of using AVRCP {raw} to pause playback:

AVRCP 46

AVRCP C6

A2DP STREAMING STOP 0

Below is an example of using AVRCP {raw} to rewind the track position:

AVRCP 48

[track rewinds until next command is sent, specific behavior depends on media device]

AVRCP C8

7 iWRAP commands related to audio

PLAY	-	Command **PLAY** is used to generate tones or beeps.
VOLUME	-	Changes the volume level (audio output gain).
SET CONTROL AUDIO	-	This command controls the physical interface routing of audio on WT32. The command is also used to enable or disable multiple SCO support.
SET CONTROL CODEC	-	This command controls the preference of A2DP audio codecs, channel modes and sampling rates for A2DP.
SET CONTROL GAIN	-	**SET CONTROL GAIN** is used to control the internal input and output gain.
SET CONTROL MICBIAS	-	**SET CONTROL MICBIAS** controls the linear regulator that drives current through the dedicated mic bias pin.
SET CONTROL PCM	-	This command configures the physical PCM hardware interface settings and PCM data format.
SET CONTROL RINGTONE	-	Configures a ring tone which is used with HFP or HSP profile if the Audio Gateway does not provide an in-band ring tone.
SET CONTROL PREAMP	-	Enables or disables the 20dB microphone pre-amplifier.

Please refer to iWRAP user guide for exact documentation of the listed commands.

NOTE:

The pre-amplifier MUST be turned off in A2DP source device to prevent the low frequencies being cut out.

8 Power saving

iWRAP offers two power saving options. Sniff mode, which can be used to save power for active *Bluetooth* connections and deep sleep more which puts the internal processor into a reduced duty cycle mode. Please refer to iWRAP user guide for more information about sniff and deep sleep modes. One should be very careful when using sniff mode with an active A2DP connection. Sniff mode will reduce the data transmission rate and therefore affect the robustness of an audio link. When sniff mode is used with A2DP audio “clipping” is more likely to occur then without power saving.

One should also know that when *Bluetooth* connections are in active mode i.e. no power saving in use the master device uses 3-4 times less power then a slave device. Therefore for battery powered applications it might be useful to configure the device as a master rather then a slave.

9 Audio quality improvement

The sub-band coding (SBC) used to encode or decode the A2DP audio is a lossy audio codec and some of the audio data is lost when it's transmitted over a *Bluetooth* link. For some applications like high end headsets and headphones or audio amplifiers this may not be acceptable.

Alternative audio codecs can be used to encode or decode the audio stream and provide better quality or lower latency *Bluetooth* audio. Such codes are for example MP3, AAC, aptX or aptX Low Latency. MP3 codec is however not generally supported by *Bluetooth* enabled cell phones, tablets and PCs. AAC codec is supported in the latest generation of Apple devices. aptX codec on the other hand has received the widest market adoption and is supported by my many cell phones, tablets, PCs and headsets.

AAC codec support is available by default from iWRAP6 onwards in all firmware builds (and also upon request for iWRAP5 audio firmware builds). However, the usage of the codec in the end product requires a separate license and it has an extra license fee. For more information, please see http://www.vialicensing.com/licensing/aac-fees.aspx.

aptX codec provides a nearly lossless audio quality and therefore is ideal to be used in Hi-Fi audio applications and products where high quality *Bluetooth* audio is needed. The highest quality of *Bluetooth* audio is provided by aptX Low Latency codec that significantly decreases the Bluetooth latency. iWRAP supports aptX and aptX Low Latency codecs, but because they are 3rd party codecs and have an extra license fee, they're not included in the generic firmware builds. For more information about aptX and aptX Low Latency codecs, please contact https://www.silabs.com/about-us/contact-sales.

9.1 Enabling AAC Audio Codec for A2DP

One of the optional audio codecs with Bluetooth A2DP is AAC. AAC is supported at the moment by Apple iOS devices and provides better audio quality compared to the regular SBC codec. WT32i supports the AAC audio codec and it can be enabled with the following iWRAP commands.

```
SET CONTROL CODEC AAC JOINT_STEREO 44100 0
SET CONTROL CODEC SBC JOINT_STEREO 44100 1
```

The first command enables the AAC codec in stereo mode and 44.1kHz sampling rate and priority 0 (highest). The second command enables the mandatory SBC codec in stereo mode and also 44.1kHz sampling rate and priority 1 (lower). With this configuration AAC is the preferred codec, but if the remote device does not support it SBC will be used instead.

Note: iWRAP contains AAC technology which incorporates intellectual property owned by numerous third parties. Supply of this product does not convey a license under the relevant intellectual property of those third parties nor imply any right to use this product in any finished end user or ready-to-use final product. An independent license for such use is required. For details, please visit http://www.vialicensing.com.

9.2 Enabling aptX Audio Codec for A2DP

aptX is another optional audio codec for Bluetooth A2DP. aptX is supported by newer Android devices and it also provides better quality audio compared to the regular SBC codec. The feature is available only for WT32i modules. The aptX audio codec can be enabled in special builds with the following iWRAP commands.

```
SET CONTROL CODEC APT-X JOINT_STEREO 44100 0
SET CONTROL CODEC SBC JOINT_STEREO 44100 1
```

Note: aptX requires a special license and is only included in a separate iWRAP build. To enable it, you need to flash the aptX build and add a valid license key. Please contact Bluegiga technical support (https://www.silabs.com/support) for further information on how to obtain a license key.

9.3 Enabling aptX Low Latency Audio Codec for A2DP

aptX Low Latency is another optional audio codec for Bluetooth A2DP. It offers a Bluetooth latency of approximately 40 ms (far less than the standard Bluetooth latency) and meets the recommended latency for audio applications. aptX Low Latency feature supports A2DP Source mode and is available only for WT32i modules.

The aptX Low Latency audio codec can be enabled in special builds with the following iWRAP commands:

```
SET CONTROL CODEC APT-X_LL JOINT_STEREO 44100 0
SET CONTROL CODEC APT-X JOINT_STEREO 44100 1
SET CONTROL CODEC SBC JOINT_STEREO 44100 2
```

Note: aptX Low Latency requires a special license and is only included in a separate iWRAP build. To enable it, you need to flash the aptX Low Latency build and add a valid license key. Please contact Bluegiga technical support (https://www.silabs.com/support) for further information on how to obtain a license key.

9.4 Routing the A2DP Audio to I2S (External Codec)

iWRAP6 adds basic support for configuring an external I2S audio codec without requiring an external MCU. The commands controlling this are "**SET CONTROL AUDIO**" and "**SET CONTROL EXTCODEC**".

The WT32i development kit contains an external I2S audio codec (Texas Instruments TLV320AIC32IRHB), which connects to the WT32i's I2S interface as well the I2C interface. To try out the external codec, the following iWRAP configurations are needed.

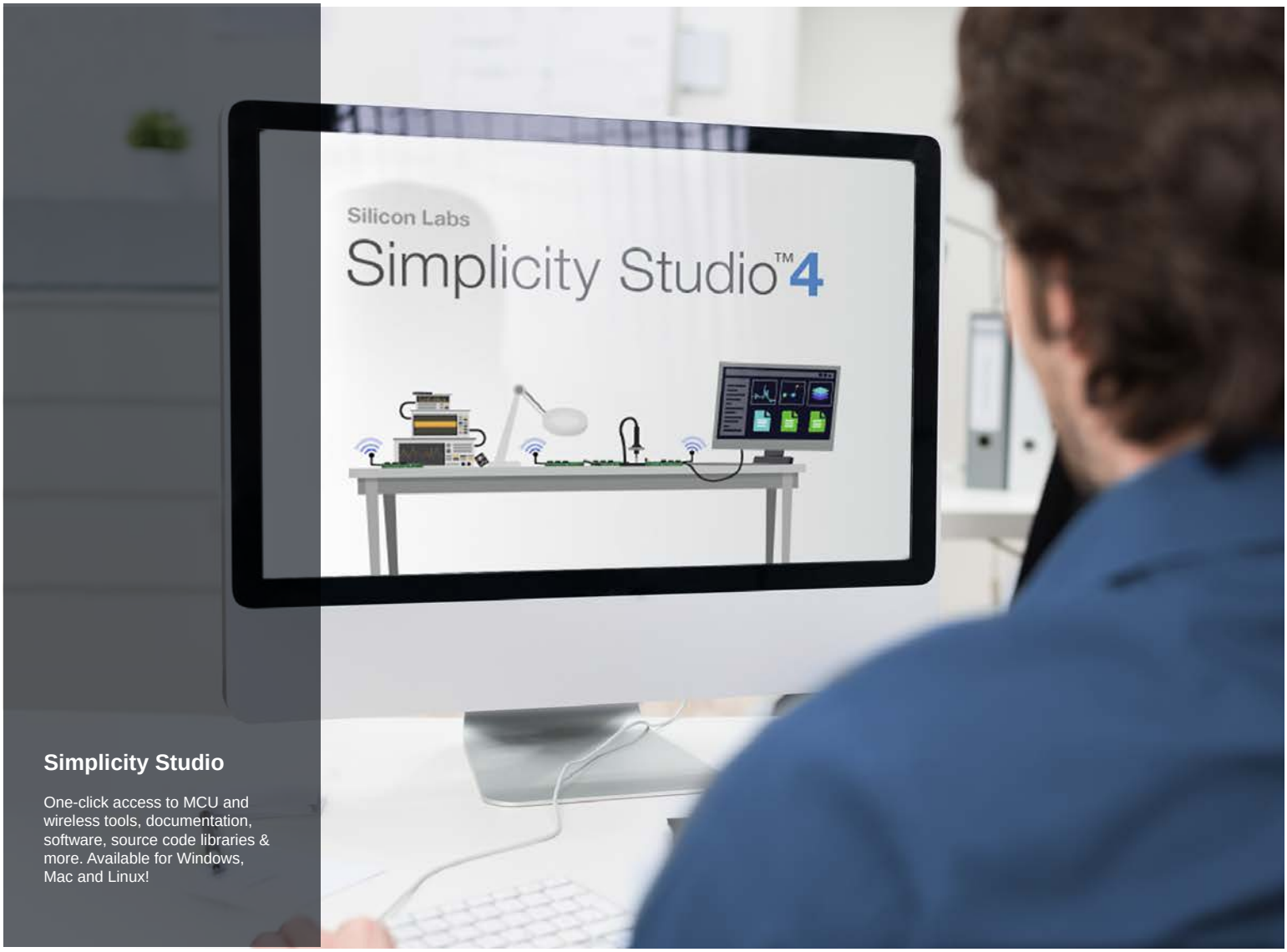
```
SET CONTROL AUDIO INTERNAL I2S EVENT 32
SET CONTROL EXTCODEC PRE 30 18 0200918000008A10370001000080000000FF07C7C787C7C78 30 04 25C01400 30 0a 28C00000000000008000 30 03 330F00 30 05 3F00800F00 30 04 6500A200
RESET
```

The first command routes the A2DP audio to the I2S interface instead of the DAC and configures I2S to use 32 bits per sample to suit the clocking requirements of the TI codec. The second command contains the codec configuration parameters that iWRAP send to the TI codec over I2C interface when A2DP streaming is enabled or disabled.

10 References

[1] The *Bluetooth* SIG, A2DP Profile overview, URL: https://www.bluetooth.com/specifications/profiles-overview

[2] The *Bluetooth* SIG, AVRCP Profile overview, URL: https://www.bluetooth.com/specifications/profiles-overview



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com