ARCAM Avatar Messaging

Draft Version v0.51

Please Note:

This document is intended as the starting point for a fully defined API between ARCAM devices and the Avatar running Stream Unlimited firmware.

The aim is to make the most of the current Hostlink codebase using existing messages where available. At this stage, the document makes an initial proposal for the API, and identifies the data items that need to be exchanged. It is expected that this will evolve during the review process.

If on review, it is found that there are more efficient or alternative existing mechanisms for exchanging the data, then the API will be amended to reflect this.

- Anything in RED is for discussion and is to be removed or addressed in the final version...
- Draft 1 assumes that we do not need to stick to the Citation API, as it does not cover the functionality required by ARCAM. If for any reason it makes sense to retain the Citation API in places, then the document will be updated to reflect this.
- This document DOES NOT cover the interaction between the Avatar and the ARCAM app, which will be defined in a separate document.

ARCAM
The West Wing
Stirling House
Waterbeach
Cambridge
CB25 9PB
United Kingdom

Version

Version	Date	By	Description
0.1	08/11/2018	Chris Mann	First draft following the meeting between ARCAM and Stream Unlimited 7-8/11/2018
0.2	16/11/2018	Chris Mann	Updated and revised based on comments on first draft from Stream Unlimited. • Handshaking changed to be similar to Citation approach. • IP Address moved to its own Status message. • Status messages for Hostname and MAC Address added • Payloads defined for a number of messages • Notes updated.
0.3	22/11/2018	Chris Mann	Updated and revised based on comments on second draft from Stream Unlimited. • Changed handshaking so that ARCAM request info rather than Avatar sending it. • Handshaking message content detailed. • Device to be responsible for setting up Avatar state. • Upgrade section revised to include User/Controlled option • Solo Meta removed This version was sent to stream 20/12/2018
0.4		Chris Mann	Version prepared for ANAME
0.5	10/01/2019	Chris Mann	Added messaging diagram
0.6	11/01/2019	Chris Mann	• Updated Device Model identifiers (Changed to 0XA rather than 0X0) • Handshaking section reworked to take into account messages observer for the Citation module • Current Sample Rate – Note added that this does not appear to be implemented in the Citation. • Configuration section revised and simplified. • Control section expanded, and Player Control message added. • Feedback/Status section reorganised and messages tidied. MAC Address and Hostname removed.

			• Browsing. Messages renamed to Browse… and expanded, Page Results replaced with generic Response message.
			• TCP. Client Connected/Disconnected replaced by a single message.
			• Control4 messages expanded.
			• Crestron changed to have single message with response rather than 2 messages.
			• Dirac. Messages expanded.
			• Power. New questions on Avatar Standby state added.
			• Upgrade.
			  ○ Upgrade Available merged into Upgrade Status.
			  ○ Upgrade Start removed.
			• Factory Reset added.

0.7	31/01/2019	Chris Mann	Changes to Crestron message to extend functionality to inform when Configuration is complete.
0.8	01/02/2019	Chris Mann	• Clarification of MAC Address data structure. • Correction of Upgrade metadata MD5 checksum size (now 16 bytes) • Addition of Audio Input Feedback message to allow the Avatar to inform the Device when the input is changed internally rather as a response to the Audio Input command.

0.9	06/03/2019	Chris Mann	• Changes applied to TCP/IP messages following comments from Peter S
			  ○ New Message Id pair added (0x0008, 0x8008)
			  ○ Client Connection message encoding changed to UINT16
			  ○ TCP Message Received and TCP Send Messages combined into TCP Client Message
			  ○ Transfer State enum identified – SU to provide values

0.10	08/03/2019	Chris Mann	• Clarification of the Offset in TCP Client Message • Clarification of byte ordering in TCP Client Connection
0.11	11/03/2019	Chris Mann	• Removal of Offset from TCP Client Message at Streams Request, replaced by maximum data

			size allowing us to calculate offset from packet_id/control_flags. • Changes to TCP Message State Ids – to use new numbering range 0x0001 -> 0x0*nnn* at request of Stream Unlimited • Addition of TCP Client Control Message
0.12	13/03/2019	Chris Mann	• Updated Networking section to confirm the TCP Client Connection response to a TCP Client Command (Disconnect) message.
0.13	21/03/2019	Chris Mann	• Added Current MIME Type message (0x0010)
0.14	29/03/2019	Chris Mann	• Updated with Matjaz's response to the Handshaking/Control/Network and Power Questions ○ Power Notes/Questions ○ System Booted Notes ○ Handshaking process ○ Firmware version number formats ○ Hostname Format ○ I2C Address 0x05 messages ○ MAC Address
0.15	02/04/2019	Chris Mann	• Removal of 0xF301 – Device Model state as no longer required (replaced by 0xF300 Device System Info) • Remove misleading text in Browse Item (0xF302) • Removal of Current Sample Rate (0x0011) from Feedback section. • Addition of Sample Rate message (0x0308) in both Control and Feedback sections. Note: This now replaces the Current Sample Rate (0x0011) in Feedback. • Re-arrange feedback section to get messages in State ID order
0.16	03/04/2019	Chris Mann	• Changes to Dirac messages to enable Dirac selection on a per Input basic (rather than globally) ○ Reduced number of curves from 5 to 3 ○ Added curve number into Calibration Status message • Additional notes added to clarify curve selection scenarios. • Fixed length padded field for Dirac Curve names.

			• Info on when the Sample Rate message is to be sent from the Avatar to the Device. • Addition of Crestron and Control4 Discovery Active messages
0.17	10/04/2019	Chris Mann	• Removal of Other Messages from Feedback as IP Address has been added to State Change messages (in a previous version) • Addition of Track Position message (0x000A) to Feedback section (for use with ST60)
0.18	23/04/2019	Chris Mann	Following an email from Matjaz 11/04 • Added TCP Transfer States • Updated TCP Transfer response requirement
0.19	23/04/2019	Chris Mann	• Addition of Factory Reset (0xF301) message in the Control section to replace the Citation Key Press - Reset Long Press Event • Change to TCP State IDS to reflect values passed by Stream.
0.20	24/04/2019	Chris Mann	• Previous Factory Reset addition (v0.19) duplicated existing Factory Reset message – now corrected so that there is only one message 0xF301 • Split Sample Rate message (0x0308) into two messages (0x0309, 0xF309) after discussion with Matjaz. Changed Upgrade States (0xF309, 0xF30A) to (0xF30A, 0xF30B) to accommodate changes.
0.21	26/04/2019	Chris Mann	• Added a Response to the TCP Client Control message to clarify the response from the Avatar
0.22	09/05/2019	Chris Mann	• Added note on URLs in Album Art • IP Address fixed as State Change message • Changed Chromecast App Name to a State Change message at Matjaz request.
0.23	15/05/2019	Chris Mann	• Update to Device System Info enum names to include hyphens in Lexicon products. • Addition of existing Hostlink Track Length feedback message for ST60
0.24	22/05/2019	Chris Mann	• TCP Sequence diagram updated for correct State IDs • Album Art ○ Album Art message changed to 0x0308

			○ New Album Art URL message added (0x0307) to allow for the caching of Album Art • Addition of Maestro and Concert models to the Device Model List
0.25	07/06/2019	Chris Mann	• Corrections to Control/Feedback sequence diagrams. • Addition of SSID State Change Message (0x030F) to the Feedback section
0.26	01/07/2019	Chris Mann	• Expansion of Manual Upgrade section • In Device System Info ○ Dropped Lexicon RV-7 ○ Changed Lexicon RV-10/MC-16 to JBL Synthesis SDR-35/55 • Changes to Device Firmware Data Transfer process to include a response from the device after each Upgrade Data message • Device Upgrade Status 0x00 Changed from Receiving Data to Ready to Receive • Avatar Upgrade Status 0x06 Device Upgrade Starting removed. • Addition of Avatar Upgrade Status 0x03 Start Upgrade and refactoring of other status values.
0.27	15/07/2019	Chris Mann	• Upgrade logic change so that a Device Upgrade Status 0x02 Transfer Successful message is sent once the final packet has been received rather than a combination of a 0x01 Data Received and a 0x02 Transfer Successful. • Removal of the 0x030B Avatar Upgrade Status 0x07 Transfer Successful state as it is no longer required.
0.28	15/07/2019	Chris Mann	• Changes to the Album Art message 0x0308 so that the response containing the Image data is an asynchronous 0x0001 Set states message.
0.29	17/07/2019	Chris Mann	• Redefines 0x0302 MAC Address to be the Ethernet MAC Address • Added 0x0310 WiFi MAC Address message • Added 0x0311 Friendly Name message • New MAC Address and Friendly Name added to Avatar System Info exchange • Added an MQA State Change message (0x0312)

0.30	25/07/2019	Chris Mann	• Addition of Avatar Upgrade Status to Handshaking
0.31	31/07/2019	Chris Mann	• Album Art URL re-defined as an SHA256 Hash • Changed Album Art Response to include Album Art URL on Filips suggestion. • Changed AlbumArt request to a SetStates message (rather than a GetStates) • Change Album Art packet size to 4KB • Friendly Name limited to 30 bytes • Current Network State message added to address issue where the network is forgotten by the user.
0.32	28/08/2019	Chris Mann	• Offset calculation removed from Album Art as incorrect. • Addition of Device Upgrade Jump To Offset field in Device Upgrade Status • Added Set Region command
0.33	03/09/2019	Chris Mann	• Added new States for Serial Number (0x0314) and MAC Address (0x0315) with Get/Set messages
0.34	04/09/2019	Chris Mann	• Added a Get Region message for consistency
0.35	10/09/2019	Chris Mann	• Changed Set MAC Address (0x0315) to use string format rather than 16-bit binary
0.36	16/09/2019	Chris Mann	• Changed data type of Device Upgrade Jump To Offset in Device Upgrade Status to UINT32 for consistency
0.37	18/09/2019	Chris Mann	• Dirac Curve name length changed to 40bytes • Dirac: Added Calibration Updated, and Removed Calibration Saved/Removed.
0.38	19/09/2019	Chris Mann	• Changed Region message to work as both a State Change and a Get States message.
0.39	04/11/2019	Chris Mann	• Added a section to explain the difference in MAC Addresses. • Clarification of TCP Message Data Length and Client Id usage • Upgrade Status message changes after Matjaz email 04/11/2019
0.40	06/11/2019	Chris Mann	• Upgraded status 0x07 Start Avatar Upgrade and 0x08 Start Avatar USB Upgrade moved to Device Upgrade Status from Avatar Upgrade Status.

			• Upgrade diagrams updated.

0.41	26/11/2019	Chris Mann	• Changes to upgrade approach applied.
			  ○ Upgraded status 0x07 Start Avatar Upgrade removed.
			  ○ Upgrade Option removed

0.42	18/12/2019	Chris Mann	• New Dirac messages added for AVR Dirac handling.
			  ○ Dirac Speaker Configuration
			  ○ Dirac Channel Volume
			  ○ Dirac Stimuli Generation
			  ○ Dirac Filter Info

0.43	21/01/2020	Chris Mann	• Added 0x0319 Dirac Client Status message defined by Matjaz. • Changed to 0x0318 Dirac Filter Info message to use UINT16 field (see Matjaz email 14/01/2020 13:44)
0.44		Chris Mann	• Re-added AVATAR Upgrade Option message for supressing OTA's see SSDKARCAM-96
0.45		Chris Mann	• Added new Display Sample Rate message (0x0329) for displaying the original Sample Rate in MQA audio.
0.46		Chris Mann	• Addition of Headphone Status message (0x0320)

0.47		Chris Mann	• Added new products into Device System Info (0xF300)
			  ○ Mark Levinson No5707 (their all in one)
			  ○ JBL HDi Active (some new active speakers based on an uno)
			  ○ JBL SA750 (the retro SA30)
			  ○ Mark Levinson NoXXXX (as yet undefined ML SA30 variant)

0.48	26/06/2020	Chris Mann	• Added MQA Decoder level message (0xF30F) • Added Lip Sync delay (0xF310)
0.49	20/07/2020	Chris Mann	• Browse Item/Browse page replaced by Browse Folder and Browse Play • Change to Browse Back (0xF304) message to include the folder name.
0.50	07/10/2020	Chris Mann	• 0xF311 – Maximum Volume message added • 0xF310 Lip Sync Delay changed to a UINT16

0.51	24/11/2020	Chris Mann	• Added entries to the Device System Info document for the KBL L75MS and the JBL 4350P

Contents

Overview

This document describes API that will be used to pass messages between ARCAM products and the Avatar running Stream Unlimited firmware. The API will form a super-set of the functionality required for each product.

At the time of writing the ARCAM products that will use the API are as follows:

- Solo Uno
- SA30
- ST60
- New AVR

The document splits the API into areas of functionality, and provides a brief overview of each functional area.

Terms

The following terms are used through the document:

Term	Description
Avatar	The Avatar board running Stream Unlimited firmware.
Device	The ARCAM device that is communication with the Avatar
Message	The mechanism by which information is transferred between the device and the Avatar and visa-versa.
Hostlink Protocol	A lightweight protocol used to communicate simple data structures between a master microcontroller running StreamSDK and the device.

References

Title	Location
Stream Unlimited Hostlink protocol	
HK Citation MCU AML Communication_2.7	
HostLink protocol_ I2C and SPI Hostlink Examples	

API Scope

For this document the API is split into a number of functional areas:

Functional Area	Description
Handshaking	Describes the interaction between the Avatar and the device during initial start-up.
Configuration	Depending on the features the device implements it may need to subscribe or unsubscribe from messages sent by the Avatar.
Control	The device will be able to control run-time aspects of the Avatars functionality such as volume / mute etc.
Feedback/Status	Because the Avatar can be controlled externally by the ARCAM app, messages are required to inform the device of the current state for display purposes
UPnP and USB Browsing	It will be possible to select music from file hierarchies stored either via UPnP or on a USB device accessed by the Avatar. This area describes the message required to pass the current browsing state from the Avatar to the device, and for the device to navigate and select individual folders and tracks.
Pre-sets	It will be possible for the user to store pre-sets in the Avatar. This area covers pre-set selection and definition.
Network	Describes the messages required to implement limited TCP tunnelling through the Avatar to the device.
Control4 / Crestron	The Avatar will implement Control4 and Crestron auto-discovery. To achieve this it requires user-interaction messages to be passed from the device.
Dirac	The Avatar will include Dirac room correction. The device will be able to choose which EQ curve should be used, and receive information on Dirac configuration status, and curve names.
Power	The device will notify the Avatar of changes in its power state.
Upgrade	Interaction is required between the device and the Avatar during the upgrade process.

	Solo Uno	SA30	ST60	AVR
Handshaking	✓	✓	✓	✓
Control	✓	✓	✓	✓
Feedback/Status		✓	✓	✓
Browsing		✓	✓	✓
Pre-sets		✓	✓	✓

Network	✓	✓	✓	✓
Control4 / Crestron	✓	✓	✓	✓
Dirac		✓		
Power	✓	✓	✓	✓
Upgrade	✓	✓	✓	✓

API FUNCTIONAL AREA USAGE BY DEVICE

General principles

The following general principles apply across all the API's functional areas.

Communication

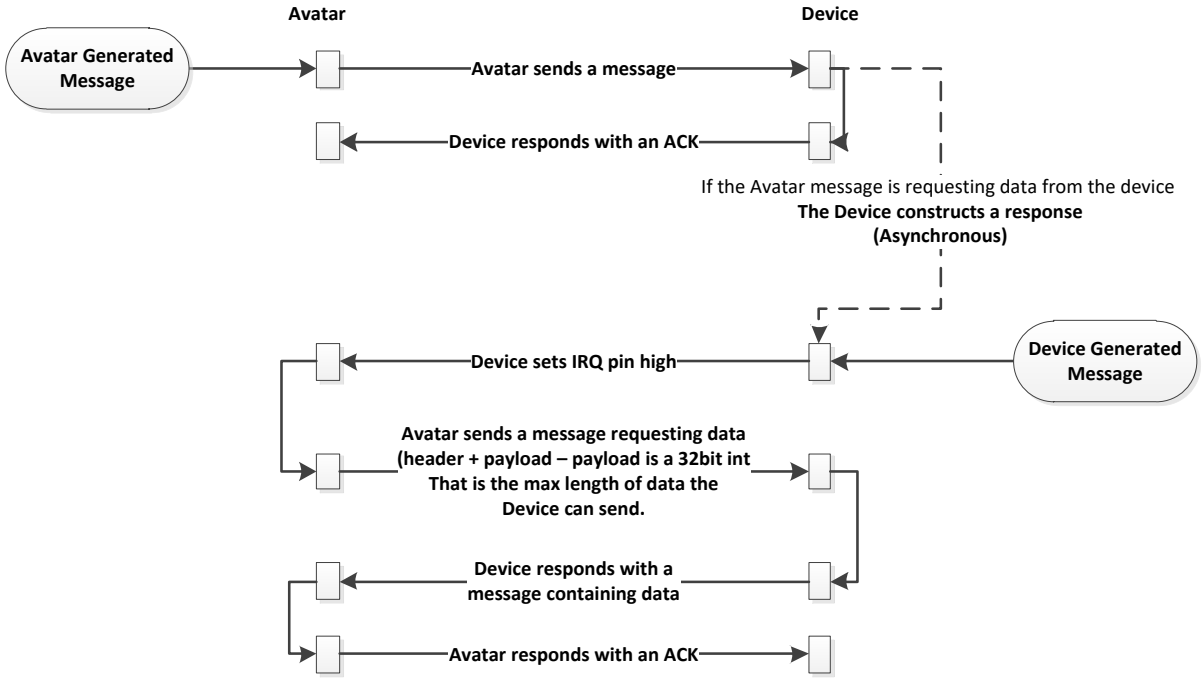
Communication between the Avatar and the Device is over I2C. The Avatar is the master and the device the slave.

The device requests communication with the avatar by pulling the IRQ line down. The avatar responds with a message containing an integer identifying the amount of data that can be transferred.

Messaging

Message flow

All communication between the Avatar and the Device follows the sequence below.



Message structure

- All messages follow the Link Packet Layout described in the Hostlink protocol API document.
  - All messages begin with a standard 8-byte Hostlink header.
  - If payload data is included in the message it will be terminated by a payload data tail (0x00)

Byte	Name	Description
0	Start of packet (LSB)	Start of packet indicator
1	Start of packet (MSB)	-
2	Packet Length (LSB)	Packet length includes -
3	Packet Length (MSB)	Header + (opt. Payload Data + Payload Data tail)
4	Control Flags	Packet exchange control bit flags
5	Frame ID	Frame number the packet belongs to

6	Packet ID	Packet number within a frame
7	Reserved	Currently set to zero
8..N	Payload Data	Optional. Payload data
N+1	Payload Data tail (LSB)	Only present if Payload Data is present
N+2	Payload Data tail (MSB)	Only present if Payload Data is present

Link Packet Layout[1]

- All messages are responded to with an ACK or NACK message which consists solely of a standard Hostlink header with the appropriate Control Flag set.
- The device initiates a message dialog by pulling the IRQ pin down. The Avatar responds by sending a message with a 4-byte payload identifying the maximum size of the message that can be sent. This messages is then responded to with the device message.

External communication

- Communication between the device and Music Life (or any other ARCAM app) will be through TCP Tunnelling.

ARCAM Specific States

A number of ARCAM specific states are defined below:

Avatar - > Device

States

State ID	Description
0x0301	IP Address
0x0302	Ethernet MAC Address
0x0303	Host Name
0x0304	Chromecast App Name
0x0305	Audio Input
0x0306	Dirac Configuration Status
0x0307	Album Art URL
0x0308	Album Art
0x0309	Sample Rate
0x030A	Crestron Configure
0x030B	Avatar Upgrade Status
0x030C	Upgrade Data Size
0x030D	Upgrade Data Offset

[1] Taken from the Hostlink API document

0x030E	Upgrade Data
0x030F	SSID
0x0310	WiFi MAC Address
0x0311	Friendly Name
0x0312	MQA State
0x0313	Region
0x0314	Serial Number
0x0315	MAC Address
0x0316	Dirac Channel Volume
0x0317	Dirac Stimuli Generation
0x0318	Dirac Filter Info
0x0319	Dirac Client Status
0x0320	Headphone Status
0x0329	Display Sample Rate

TCP States

State ID	Description
0x0000	TCP Client Connection
0x0001	TCP Client Message
0x0002	TCP Client Control

Key Events

Key ID	Description

Device -> Avatar

States

State ID	Description
0xF001	Host Firmware Version (Existing HostLink message)
0xF300	Device System Info

0xF301	Factory Reset
0xF302	Browse Item
0xF303	Browse Page
0xF304	Browse Back
0xF305	Control4 Discovery Active
0xF306	Control4 Identify
0xF307	Dirac Curve Names
0xF308	Dirac Select Curve
0xF309	Set Sample Rate
0xF30A	Upgrade Option
0xF30B	Device Upgrade Status
0xF30C	Crestron Discovery Active
0xF30D	Device Upgrade Jump To Offset
0xF30E	Dirac Speaker Configuration
0xF30F	MQA Decoder Level
0xF310	Lip Sync delay
0xF311	Maximum Volume

Key Events

Key ID	Description

Questions / Notes

While defining the messages between the Device and Avatar it appears that there is no need for any Key Events, as all communication can be handled by Get States, Set States or State Change messages – does that sound OK?

It is also possible that the allocation of State Ids in the 0x0300 and 0xF300 ranges may not follow Stream Unlimited standards – these can be changed if necessary.

MAC Addresses

There are three states defined for MAC Addresses:

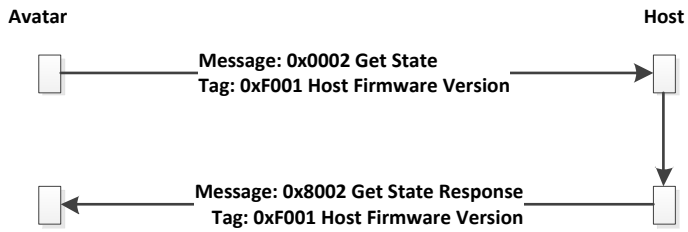
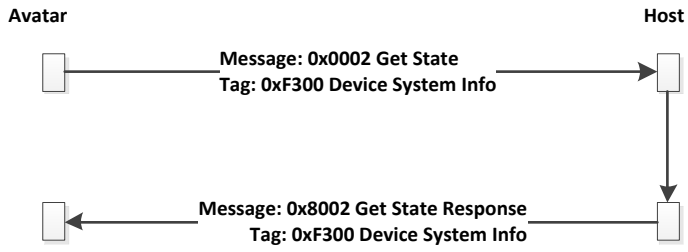
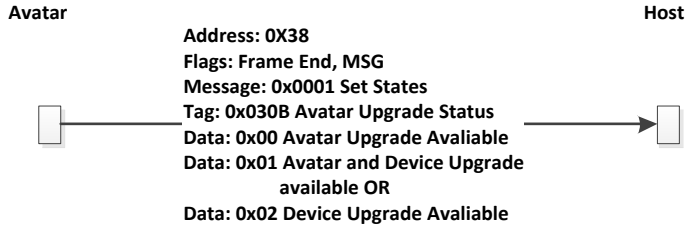
- 0x0302 – Returns a UINT16 of the currently used Ethernet MAC Address.
- 0x0310 – The current WiFi MAC Address
- 0x0315 – Returns a MAC string that is stored on NAND

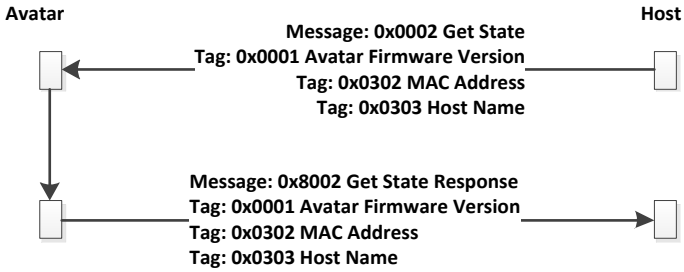
In principle, 0x0302 and 0x0315 should usually be the same, but for example, on a module that has had no MAC set yet, 0x315 will return nothing, and 0x302 will return the currently used randomly generated MAC. Similarly, on a unit that just had a new MAC value configured, 0x315 will already reflect the new value, while 0x302 will still report the previous address until the device is actually rebooted.

Handshaking

The handshaking sequence will be performed before any other transfer of data between the Avatar and the device. The sequence is initiated by the Avatar.

The diagram below identifies the messages sent between the Avatar and the Device. Note: The diagram only shows the main messages in the sequence and for clarity does not show ACKs, and IRQ messages, however they all follow the messaging sequence described in the General Principles section.

Host Firmware Version 	Handshaking is initiated by the Avatar sending a request for the Devices Firmware Version
Device System Info 	The Avatar requests system info from the device
Avatar Upgrade Status 	The Avatar reports on whether there is an upgrade available (Optional)

System Ready  	The Avatar informs the Device that it has completed its boot sequence
Avatar System Info  	The Device requests system info from the Avatar

Host Firmware Version

Message	Host Firmware Version
	The Avatar requests the Device Firmware version

Request : Avatar->Device		
Message	0x0002	Get States
Count	1	
State	0xF001	Host Firmware Version

Response : Device->Avatar		
Message	0x8002	Get States Response
Count	1	
State	0xF001	Host Firmware Version
Encoding	0xC5	BIN16
Value		The Host Firmware version – see Notes for format.

Notes	As 3.1 of Citation document

Device System Info

Message	Device System Info
Direction	Device -> Avatar

The Avatar requests the Devices System Info which is provides the Device Model.

Request : Avatar->Device		
Message	0x0002	Get States
Count	1	
State	0xF300	Device System Info

Response : Device->Avatar		
Message	0x8002	Get States Response
Count	1	
State	0xF300	Device System Info
Encoding	0xCD	UINT16
Value		The value returned identifies the Device Model, allowing the Avatar to select device specific settings. E.g. Device Model: 0x00A0 ARCAM Solo Uno 0x00A1 ARCAM SA30 0x00A2 ARCAM AVR10 0x00A3 ARCAM AVR20 0x00A4 ARCAM AVR30 0x00A5 ARCAM AV40 0x00A6 *not used* 0x00A7 JBL Synthesis SDR-35 0x00A8 JBL Synthesis SDP-55 0x00A9 ARCAM ST60 0x00AA Audio Control Maestro X7 0x00AB Audio Control Maestro X9 0x00AC Audio Control Concert XR-4 0x00AD Audio Control Concert XR-6 0x00AE Audio Control Concert XR-8

			0x00AF Mark Levinson No5707 0x00B0 JBL HDi active 0x00B1 JBL SA750 0x00B2 Mark Levinson NoXXXX (SA30 Variant) 0x00B3 JBL L75MS 0x00B4 JBL 4350P
Notes	The format of the Firmware version will be *major_version.minor_version* e.g. 1.3		

System Ready

Message	System Ready		
	The Avatar indicates that it has booted and is ready to start.		
	Request : Avatar->Device		
	Message	0x0004	State Change
	Count	1	
	State	0x0020	System Booted and Comms Ready
Notes	This message signifies that StreamSDK should be up and ready to receive HostLink messages. It does not assure networking is available at this point.		

Avatar System Info

Message	Avatar System Info		
Direction	Device -> Avatar		
	This is a message pair based on the standard Get Status message request and response. The request will ask for the status of a number of items in a single message: • Avatar Firmware version • Ethernet MAC Address • WiFi MAC Address • Host name		
	Request : Device->Avatar		
	Message	0x0002	Get States

Count	5	
State	0x0001	State ID for Avatar Firmware version
	0x0302	State ID for Ethernet MAC Address
	0x0310	State ID for WiFi MAC Address
	0x0303	State ID for Host Name
	0x0311	State ID for Friendly Name

Response : Avatar->Device		
Message	0x8002	Get States Response
Count	3	
State	0x0001	State ID for Avatar Firmware version
Encoding	0xC5	Firmware version encoding (BIN16)
Value		
State	0x0302	State ID for Ethernet MAC Address
Encoding	0xC5	Ethernet MAC Address encoding (BIN16)
Value		An array of UINT8 mac_address[6]
State	0x0310	State ID for WiFi MAC Address
Encoding	0xC5	WiFi MAC Address encoding (BIN16)
Value		An array of UINT8 mac_address[6]
State	0x0303	State ID for Host Name
Encoding	0xC5	Host Name encoding (BIN16)
Value		
State	0x0311	State ID for Friendly Name
Encoding	0xC5	Host Name encoding (BIN16)
Value		

Notes

Firmware Version

For 0x0001, the Firmware Version format is the same as in Hawkbit and StreamSDK (info accessible over http://<ip>:80), a string of four integers, separated by dots, the last one being hexadecimal, like so: 0.100.312.0x77f07bb. The first integer denotes version format and is always zero. The second is StreamSDK Product ID and assigned to 100 for Arcam (97-99 for Citation). The third is the sequential build number from our automated build server; 312 means

	build #312 for this project (for production release builds, this number can be customised). The last integer is the auto-generated SHA1 hash from Streams git repository **MAC Address** *Matjaz*: The current implementation reports the current active interface's address (whether it be Wi-Fi or Ethernet), but my guess is that this might not be the desired behaviour. *Our response*: We use the MAC address for display and to generate a unique ID for the device. Given that you are handling the functionality that requires the unique names (Networking, Control4, Crestron) and that we currently only request it during handshaking, then we are probably OK with being given whichever one you regard as active at the time we request it. 17/07/2019 – Changed with the addition of the WiFi MAC Address message (0x0310) **Hostname Format** *Matjaz* - At the moment, the hostname is in this format: a113dArcam-02c9114a0911 (build name + WiFi MAC address); this is too much for 16 bytes, but can be changed to anything as long as it stays unique per device (writing from top of my head, it's a requirement for some of the components like AirPlay). *Our response*: Our only requirement is that the hostname is a reasonably small length (10-16bytes) and as unique as possible. As we are working with a VFD we need to be able to display it in 16 characters, though this is for info only. In the past we have used things like the Model name plus a sub-section of the MAC address.

Questions / Notes

• This version proposes a Handshaking sequence based upon the sequence observed during the start-up of the Citation module. However, there are some differences…
• At the moment the Citation version of the Avatar outputs messages whether the preceding messages are acknowledged or not. Is it possible to ensure the handshaking process is complete before sending other messages?
  ○ *Matjaz*: *In theory yes. We'll have to test to make sure no regressions come out of it; StreamSDK is quite modular and different messages may be sent from different modules asynchronously. We'll see if blocking messages until handshaking is complete results in any strange behaviour.*
  ○ *Our response: At the moment I think this only has significance around the upgrade process where we can find ourselves in a position where the Host reports a new firmware version before having had the chance to send an Upgrade Complete message (see the Upgrade section of the document). In that case I think it will be useful to be clear about where we are in the Handshaking process, though probably not the end of the world if we can't.*

- The Citation MCU HW Info (0x0200) is replaced by the Device System Info message (0xF300)
- We have observed a Get State (0x0002) – MCU HW Info (0x0200) message being sent by the Avatar to address 0x05 before any messages to address 0x38. It appears that unless this is responded to the Avatar is not discoverable after its initial start-up. We would rather not have to handle I2C messages directed at 2 addresses if possible – is this message required given that we include a System Info message later in the process?
    - This is apparently a bug.
- At the end of the handshaking process the Avatar will be in a position to start processing audio signals, and is ready to be configured, but with non-technical caveats due to Google Cast. Usually, Google doesn't allow audio output until configuration with the Google Home app has been completed.

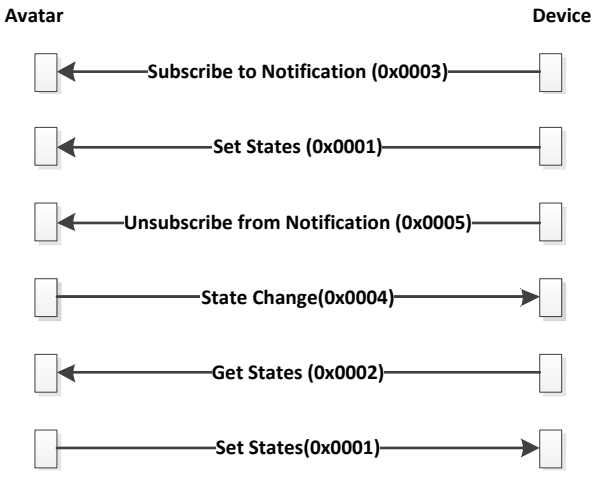
The Subscribe to Notification message is covered in the Configuration section below.

Configuration

Once the handshaking phase is complete the device will perform any configuration activities as required. Depending on the device type, Configuration may include the following:

- Subscribing to change events in the Avatar
    - Artist/Track/Album details
    - Player Status
    - Volume
    - Network Status
    - Sample Rate
- Dirac configuration
    - EQ selection
- Input Configuration
    - Avatar Internal/External input.

To achieve this the Device will send the Avatar a combination of standard Set States (0x0001), and Subscribe Values (0x0003) messages. Where the Device needs information from the Avatar we will either use a Get States or Set States message (TBD)



The list of States that may be subscribed to for notifications is described in the Feedback / Status section.

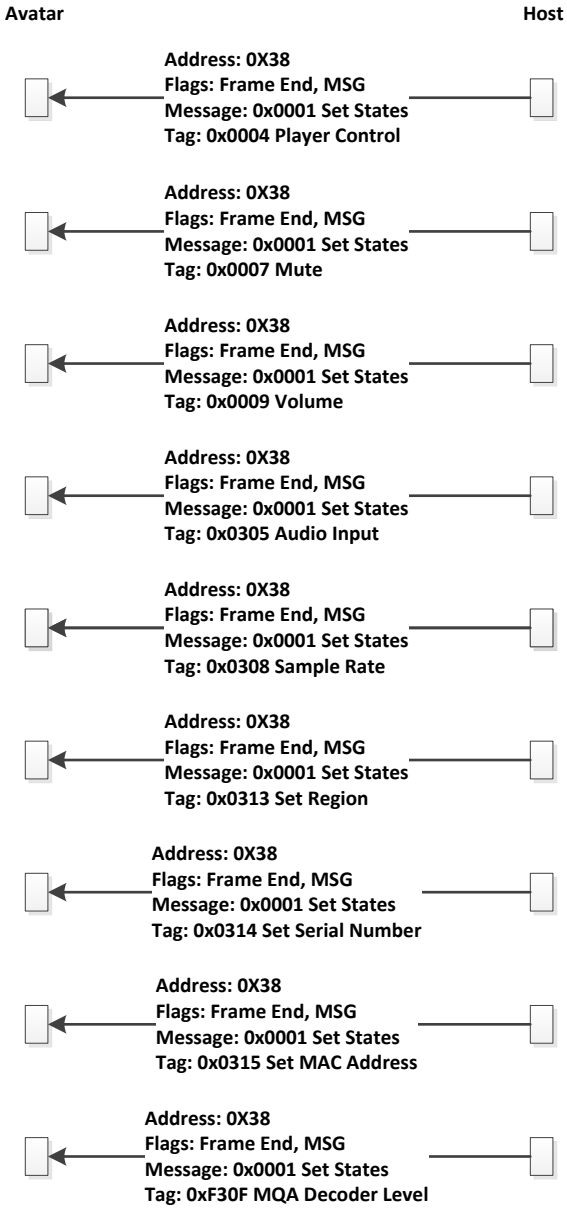
It may be necessary to unsubscribe from State Change notifications in which case the standard Hostlink Unsubscribe message (0x0005) will be used.

Questions / Notes

- We will subscribe to the states we want notification on as part of the Configuration process that follows the Handshaking process.
- It is not envisaged that we will need to dynamically unsubscribe from messages .

Control

Control messages will be sent from the Device to the Avatar to control the current status of the Avatar. These will all use the standard Set States message (0x0001).



Player Control

<table>
<tr><td>Message</td><td>Player Control</td></tr>
<tr><td>Direction</td><td>Device -> Avatar</td></tr>
<tr><td></td><td>This message is used to control the Avatars current playback.
<table>
<tr><td colspan="3">Request : Device->Avatar</td></tr>
<tr><td>Message</td><td>0x0001</td><td>Set States</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0x0004</td><td>Player Control</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>Value</td><td></td><td>Control Type
0x00 Pause
0x01 Stop
0x02 Next
0x03 Previous
0x04 Like
0x05 Dislike</td></tr>
</table>
We may also make use of the Repeat and Shuffle State messages (0x0005 and 0x0006).
Repeat States (0x0005)
0x00 Normal
0x01 Repeat One
0x02 Repeat All
Shuffle States (0x0006)
0x00 Normal
0x01 Shuffle
</td></tr>
<tr><td>Notes</td><td>
This will be identical to the existing Hostlink Set State message using the Player Control State ID (0x0004)
A Pause will be sent when the Device Audio source changes from the Avatar to an external source (CD/AUX etc.)
In the case of the SA30 it is assumed that the Device will only switch to the Avatar when the Avatar is put into a playing state by the user (via Chromecast etc.), and that it will handle putting itself into the Playing state.
To tell the Avatar to play toggle Pause (0x00)
</td></tr>
</table>

Mute

<table>
<tr><td>Message</td><td>Mute</td></tr>
<tr><td>Direction</td><td>Device -> Avatar</td></tr>
<tr><td></td><td>
This message allows the device to request the Avatar mutes or unmutes its output. This will be used when the Device does not require output from the Avatar (such as in SA30 Direct mode)
<table>
<tr><td colspan="3">Request : Device->Avatar</td></tr>
<tr><td>Message</td><td>0x0001</td><td>Set States</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
</table>
</td></tr>
</table>

	State	0x0007	Mute State
	Encoding	0xCC	UINT8
	Value		**Mute State** 0x00 Unmute 0x01 Mute
Notes	This will be identical to the existing Hostlink Set State message using the Mute State ID (0x0007)		

Volume

Message	Volume		
Direction	Device -> Avatar		
	This message allows the Device to inform the Avatar that a Volume change has been made. This is so that the change in volume can be reflected by any devices streaming to/controlling the Avatar.		
	Request : Device->Avatar		
	Message	0x0001	Set States
	Count	1	
	State	0x0009	Volume
	Encoding	0xCC	UINT8
	Value		A value between 0 and 100.
Notes	This will be identical to the existing Hostlink Set State message using the Volume State ID (0x0009) It is expected that the Avatar will generate Volume State Change events as a result of the volume being set externally. The Device software will handle these to prevent feedback loops.		

Audio Input

Message	Audio Input
Direction	Device -> Avatar
	This message allows the device to notify the Avatar that the input selection has changed. This message will be used to indicate that the Avatar should switch its input between its internal and external inputs.

<table>
<tr><td></td><td>
<table>
<tr><td colspan="3">Request : Device->Avatar</td></tr>
<tr><td>Message</td><td>0x0001</td><td>Set States</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0x0305</td><td>Audio Input</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
<tr><td>Value</td><td></td><td>Audio Source. The Avatar audio source will be either Internal or External.

• 0x00 Avatar Internal Input
• 0x01 Avatar External Input
• 0x02 Avatar Bypass Mode</td></tr>
</table>
</td></tr>
<tr><td>Notes</td><td>This will be in the same format as the Citation Select Audio Source message (0x0209) but will be a new message (0x0305) with the options limited to Internal or External as unlike the Citation it is not expected that the Avatar will need to control which output the device is using.

Avatar Bypass mode is intended for use where the Device audio path does not include the Avatar.

As the Avatar doesn’t remember the last input that was selected the Device will be responsible for setting the Avatar input during configuration.</td></tr>
</table>

Set Sample Rate

<table>
<tr><td>Message</td><td>Set Sample Rate</td></tr>
<tr><td>Direction</td><td>Device -> Avatar</td></tr>
<tr><td></td><td>This message allows the device to notify the Avatar that the input sample rate has changed.

<table>
<tr><td colspan="3">Request : Device->Avatar</td></tr>
<tr><td>Message</td><td>0x0001</td><td>Set States</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0xF309</td><td>Set Sample Rate</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
<tr><td>Value</td><td></td><td>Sample Rate.
• 0x00 - Unsupported</td></tr>
</table>
</td></tr>
</table>

			• 0x01 - 32KHz • 0x02 - 44.1KHz • 0x03 - 48KHz • 0x04 - 88.2KHz • 0x05 - 96KHz • 0x06 - 176.4KHz • 0x07 - 192KHz
Notes			

Set Region

Message	Set Region		
Direction	Device -> Avatar		
	This message allows the device to notify the Avatar that the WiFi Region has changed.		
	Request : Device->Avatar		
	Message	0x0001	Set States
	Count	1	
	State	0x0313	Set Region
	Encoding	0xCC	UINT8
	Value		Region • 0x00 - Unsupported • 0x01 - US • 0x02 - CA • 0x03 - TW • 0x04 - SG • 0x05 - DE • 0x06 – KR • 0x07 – VN • 0x08 – MY • 0x09 – PH • 0x0A – NZ • 0x0B – AU • 0x0C – AE • 0x0D – ZA • 0x0E – HR • 0x0F – JP • 0x10 – RU • 0x11 – IN • 0x12 – CN

			• 0x13 – ID • 0x14 – IL • 0x15 – BR
Notes	After this has been called the Avatar will perform a reboot (to load new drivers)		

Set Serial Number / MAC Address

Message	Set Serial Number /MAC Address		
Direction	Device -> Avatar		
	This message allows the device to set both the Serial Number and MAC Address		
	Request : Device->Avatar		
	Message	0x0001	Set States
	Count	2	
	State	0x0314	Set Serial Number
	Encoding	0xC5	BIN16
	Value		This will contain the Serial Number
	State	0x0315	Set MAC Address
	Encoding	0xC5	BIN16
	Value		This will contain the MAC address as a 16-bit integer.
Notes	Its expected that the two states will generally be included in the same message		

Headphone Status

Message	Headphone Status		
Direction	Device -> Avatar		
	This message allows the device to notify the Avatar when Headphones are connected/disconnected.		
	Request : Device->Avatar		
	Message	0x0001	Set States
	Count	1	
	State	0x0320	Headphone Status
	Encoding	0xCC	UINT8

	Value		Headphone Status • 0x00 - Disconnected • 0x01 - Connected
Notes			

Questions / Notes

MQA Decoder Level

Message	MQA Decoder Level		
Direction	Device -> Avatar		
	This message allows the device to notify the Avatar of the selected MQA Decoder Level		
	Request : Device->Avatar		
	Message	0x0001	Set States
	Count	1	
	State	0xF30F	MQA Decoder Level
	Encoding	0xCC	UINT8
	Value		MQA Decoder Level • 0x00 – Full Decoder • 0x01 – Core Decoder • 0x02 – Authenticating Decoder
Notes			

Lip Sync Delay

Message	Lip Sync Delay		
Direction	Device -> Avatar		
	This message allows the device to notify the Avatar of the delay to be used for Lip Sync		
	Request : Device->Avatar		
	Message	0x0001	Set States
	Count	1	

	State	0xF310	Lip Sync Delay
	Encoding	0xCD	UINT16
	Value		The delay to be used for Lip Synching (in microseconds)
Notes			

Maximum Volume

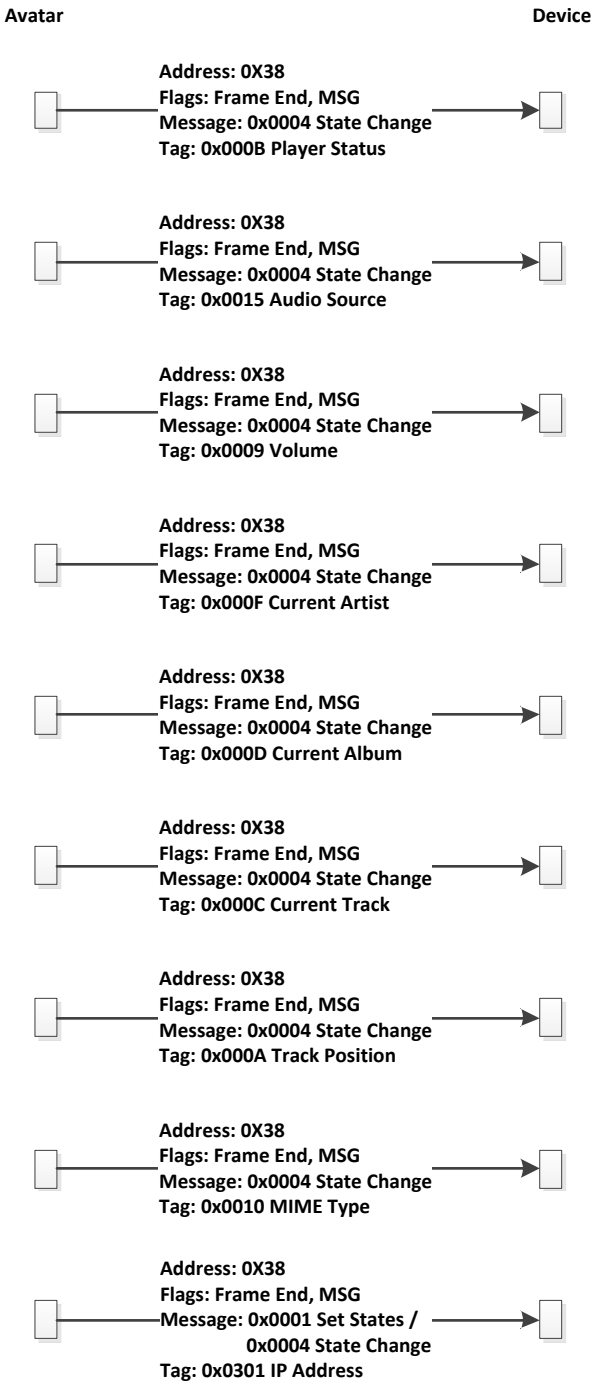
Message	Maximum Volume		
Direction	Device -> Avatar		
	This message allows the device to notify the Avatar of the Maximum Volume value		
	Request : Device->Avatar		
	Message	0x0001	Set States
	Count	1	
	State	0xF311	Maximum Volume
	Encoding	0xCC	UINT8
	Value		The Maximum Volume value 1-99
Notes			

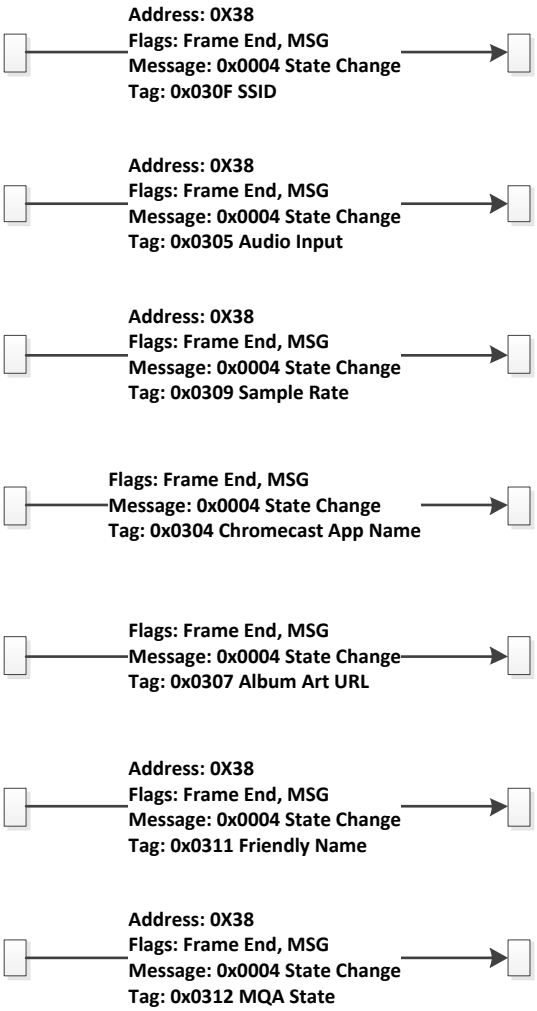
Questions / Notes

Feedback / Status

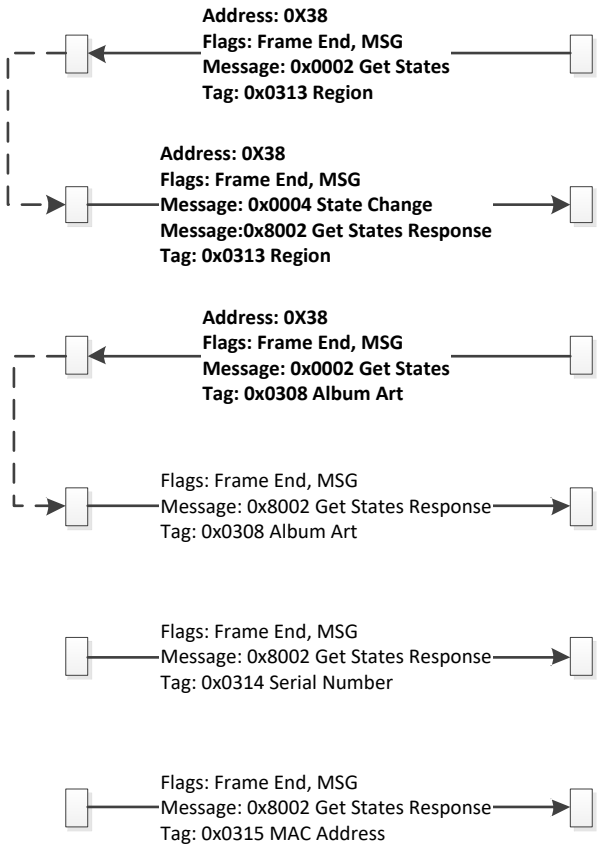
Where the device has a display, feedback on the current state of the Avatar is required. Messages are either sent from the Avatar as unsolicited Set State messages (0x0001), subscribed notifications (State Change message 0x0004), or requested from the Avatar by the device as required (using Get State message 0x0002, with a Get States Response (0x8002)).

State Change messages




Address: 0X38
Flags: Frame End, MSG
Message: 0x0004 State Change
Tag: 0x030F SSID
Address: 0X38
Flags: Frame End, MSG
Message: 0x0004 State Change
Tag: 0x0305 Audio Input
Address: 0X38
Flags: Frame End, MSG
Message: 0x0004 State Change
Tag: 0x0309 Sample Rate
Flags: Frame End, MSG
Message: 0x0004 State Change
Tag: 0x0304 Chromecast App Name
Flags: Frame End, MSG
Message: 0x0004 State Change
Tag: 0x0307 Album Art URL
Address: 0X38
Flags: Frame End, MSG
Message: 0x0004 State Change
Tag: 0x0311 Friendly Name
Address: 0X38
Flags: Frame End, MSG
Message: 0x0004 State Change
Tag: 0x0312 MQA State

Get States messages



Message Headers

The majority of the Feedback/Status messages use the standard Hostlink messages. The message descriptions below indicate the expected message type used to deliver the information.

Messages sent as notifications (Avatar -> Device) will include the following header:

Message Header		
Bytes	**Value**	**Description**
2	0x0004	The message Id for a State Change message
1	*Variable*	Number of Values included in the message
		Individual State ID data (See below)

Messages requesting State information (Device -> Avatar) will include the following header:

Message Header		
Bytes	**Value**	**Description**
2	0x0002	The message Id for a Get States message
1	*Variable*	Number of Values included in the message

		Individual State IDs (See below)

Messages sent in response to a Get States request (Avatar -> Device) will include the following header:

Message Header		
Bytes	**Value**	**Description**
2	0x8002	The message Id for a Get States Response message
1	*Variable*	Number of Values included in the message
		Individual State ID data (See below)

Player Status

Message	Player Status
Direction	Avatar -> Device
	This message allows the Avatar to notify the device of the Player Status

Request : Avatar->Device

Message	0x0004	State Change
Count	1	
State	0x000B	Player Status
Encoding	0xCC	UINT8
Value		**Player State** 0x00 Stopped 0x01 Transitioning 0x02 Playing 0x03 Paused

Notes	This is identical to the Hostlink Player Status message 0x000B

Current Audio Source

Message	Current Audio Source
Direction	Avatar -> Device

<table>
<tr><td></td><td>This message allows the Avatar to notify the device that its current Audio Source has changed.

<table>
<tr><td colspan="3">Request : Avatar->Device</td></tr>
<tr><td>Message</td><td>0x0004</td><td>State Change</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0x0015
(Note 1)</td><td>Audio Source</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
<tr><td>Value</td><td></td><td>Audio Source
• 0x00 Unknown
• 0x01 Airable
• 0x02 Multiroom
• 0x03 Files
• 0x04 Storage, CD-ROM
• 0x05 Ipod
• 0x06 VTuner
• 0x07 Rhapsody
• 0x08 Sirius
• 0x09 TuneIn
• 0x0A UPnP
• 0x0B QPlay
• 0x0C Bluetooth
• 0x0D AirPlay
• 0x0E CD-DA
• 0x0F Spotify
• 0x10 GoogleCast
• 0x11 AirableRadios
• 0x12 AirablePodcasts
• 0x13 Napster
• 0x14 Qobuz
• 0x15 Deezer
• 0x16 Tidal
• 0x17 Roon
• 0x18 AUX</td></tr>
</table>
</td></tr>
<tr><td>Notes</td><td>This will be identical to the existing Hostlink Get State message using the Audio Source State ID (0x0015). When the value is 0x10 (GoogleCast) the Chromecast App Name message can be used to identify the source</td></tr>
</table>

Current Volume

Message	Current Volume

<table>
<tr><td>Direction</td><td>Avatar -> Device</td></tr>
<tr><td></td><td>This message allows the Avatar to inform that device that it has changed its internal volume.
<table>
<tr><td colspan="3">Request : Avatar->Device</td></tr>
<tr><td>Message</td><td>0x0004</td><td>State Change</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0x0009</td><td>Volume</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
<tr><td>Value</td><td></td><td>A value between 0-100</td></tr>
</table></td></tr>
<tr><td>Notes</td><td>This will be identical to the existing Hostlink Set State message using the Volume State ID (0x0009) as a notification.

It is expected that the Avatar will send these message either when the user increases the volume on the device that they are using to control Avatar playback, or in response to a Volume message from the Device. Where the message is sent by the Device it will be the responsibility of the Device software to filter the message to prevent feedback loops.</td></tr>
</table>

Current Artist

<table>
<tr><td>Message</td><td>Current Artist</td></tr>
<tr><td>Direction</td><td>Avatar -> Device</td></tr>
<tr><td></td><td>This message allows the Avatar to inform that device that currently playing Artist has changed. This is generally included in a message packet indicating a change in Player State (0x000B)
<table>
<tr><td colspan="3">Request : Avatar->Device</td></tr>
<tr><td>Message</td><td>0x0004</td><td>State Change</td></tr>
<tr><td>Count</td><td>1</td><td>This is expected to be part of a message packet containing Player State, Artist, Album and Track states.</td></tr>
<tr><td>State</td><td>0x000F</td><td>Current Artist</td></tr>
<tr><td>Encoding</td><td>0xC5</td><td>BIN16</td></tr>
<tr><td>Value</td><td></td><td>The Current Artist Name - Blank when no track is playing.</td></tr>
</table></td></tr>
<tr><td>Notes</td><td>This will be identical to the existing Hostlink State Change message using the Artist State ID (0x000F) as a notification.</td></tr>
</table>

Current Album

<table>
<tr><td>Message</td><td>Current Album</td></tr>
<tr><td>Direction</td><td>Avatar -> Device</td></tr>
<tr><td></td><td>This message allows the Avatar to inform that device that currently playing Album has changed. This is generally included in a message packet indicating a change in Player State (0x000B)
<table>
<tr><td colspan="3">Request : Avatar->Device</td></tr>
<tr><td>Message</td><td>0x0004</td><td>State Change</td></tr>
<tr><td>Count</td><td>1</td><td>This is expected to be part of a message packet containing Player State, Artist, Album and Track states.</td></tr>
<tr><td>State</td><td>0x000D</td><td>Current Album</td></tr>
<tr><td>Encoding</td><td>0xC5</td><td>BIN16</td></tr>
<tr><td>Value</td><td></td><td>The Current Album Name - Blank when no track is playing.</td></tr>
</table></td></tr>
<tr><td>Notes</td><td>This will be identical to the existing Hostlink State Change message using the Album State ID (0x000D) as a notification</td></tr>
</table>

Current Track

<table>
<tr><td>Message</td><td>Current Track</td></tr>
<tr><td>Direction</td><td>Avatar -> Device</td></tr>
<tr><td></td><td>This message allows the Avatar to inform that device that currently playing Track has changed. This is generally included in a message packet indicating a change in Player State (0x000B)
<table>
<tr><td colspan="3">Request : Avatar->Device</td></tr>
<tr><td>Message</td><td>0x0004</td><td>State Change</td></tr>
<tr><td>Count</td><td>1</td><td>This is expected to be part of a message packet containing Player State, Artist, Album and Track states.</td></tr>
<tr><td>State</td><td>0x000C</td><td>Current Track Name</td></tr>
<tr><td>Encoding</td><td>0xC5</td><td>BIN16</td></tr>
</table></td></tr>
</table>

	Value		The Current Track Name - Blank when no track is playing.
Notes	This will be identical to the existing Hostlink State Change message using the Track State ID (0x000C) as a notification		

Track Length

Message	Track Length		
Direction	Avatar -> Device		
	This message allows the Avatar to inform that device of the current playback position in the track (in milliseconds)		
	Request : Avatar->Device		
	Message	0x0004	State Change
	Count	1	
	State	0x0014	Track Length
	Encoding	0xCF	UINT64
	Value		The current Track Length in milliseconds
Notes	This will be identical to the existing Hostlink State Change message using the Track Length State ID (0x0014) as a notification		

Track Position

Message	Track Position		
Direction	Avatar -> Device		
	This message allows the Avatar to inform that device of the current playback position in the track (in milliseconds)		
	Request : Avatar->Device		
	Message	0x0004	State Change
	Count	1	
	State	0x000A	Track Position
	Encoding	0xCF	UINT64
	Value		The current Track Position in milliseconds

Notes	This will be identical to the existing Hostlink State Change message using the Track Time ID (0x000A) as a notification

Current MIME Type

Message	Current MIME Type
Direction	Avatar -> Device
	This message allows the Avatar to inform that device of the MIME type of the currently playing track.

Request : Avatar->Device

Message	0x0004	State Change
Count	1	
State	0x0010	MIME Type
Encoding	0xC5	BIN16
Value		The value returned will be one of the following: • audio/unknown • audio/mpeg • audio/x-ms-wma • audio/ogg • audio/flac • audio/x-wav • audio/x-aiff • audio/x-pn-realaudio • audio/x-mpegurl • audio/x-scpls • application/vnd.ms-wpl • audio/mp4 • audio/x-dsd • audio/opus • audio/x-siriusxm-2

Notes	This will be identical to the existing Hostlink State Change message using the Current Playing Media MIME Type ID (0x0010) as a notification Note: The list of returned MIME types will be reviewed as we progress

IP Address

Message	IP Address
Direction	Avatar -> Device

	The message contains the IP Address assigned to the Avatar. **Request :** Avatar->Device Message: 0x0004 — State Change Count: 1 State: 0x0301 — IP Address Encoding: 0xCE — UINT32 Value: The IP Address will be passed as a 32 bit integer.
Notes	

SSID

Message	SSID
Direction	Avatar -> Device
	The message contains the SSID assigned to the Avatar. If the Avatar connection is wired it will return “Wired” **Request :** Avatar->Device Message: 0x0004 — State Change Count: 1 State: 0x030F — SSID Encoding: 0xCE — BIN16 Value: The SSID assigned to the Avatar. If the connection is wired it will return ““Wired”
Notes	

Audio Input

Message	Audio Input
Direction	Avatar -> Device
	This message allows the Avatar to notify the Device that the Audio Input has changed in circumstances where the input change has been initiated by the Avatar rather than by a SetStates message from the Device (See Audio Input message in the Control Section). **Request :** Device->Avatar Message: 0x0004 — State Change

	Count	1	
	State	0x0305 (Note 1)	Audio Input
	Encoding	0xCC	UINT8
	Value		Audio Source. The Avatar audio source will be either Internal or External. • 0x00 Avatar Internal Input • 0x01 Avatar External Input
Notes	This will be in the same format as the Citation Select Audio Source message (0x0209) but will be a new message (0x0305) with the options limited to Internal or External as unlike the Citation it is not expected that the Avatar will need to control which output the device is using.		

Sample Rate

Message	Sample Rate		
Direction	Avatar -> Device		
	This message allows the Avatar to notify the Device that the input sample rate has changed.		
	Request : Device->Avatar		
	Message	0x0004	State Change
	Count	1	
	State	0x0309	Sample Rate
	Encoding	0xCC	UINT8
	Value		**Sample Rate.** • 0x00 - Unsupported • 0x01 - 32KHz • 0x02 - 44.1KHz • 0x03 - 48KHz • 0x04 - 88.2KHz • 0x05 - 96KHz • 0x06 - 176.4KHz • 0x07 - 192KHz
Notes	A Sample Rate message will be sent whenever the Sample Rate in the avatar changes. This enabled the Device to display the sample rate when streaming as well as when receiving external input.		

Display Sample Rate

<table>
<tr><td>Message</td><td>Sample Rate</td></tr>
<tr><td>Direction</td><td>Avatar -> Device</td></tr>
<tr><td></td><td>This message allows the Avatar to notify the Device that the input sample rate has changed. Identical to the Sample rate (0x0309) message but displays the original sample rate for MQA files.
<table>
<tr><td colspan="3">Request : Device->Avatar</td></tr>
<tr><td>Message</td><td>0x0004</td><td>State Change</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0x0329</td><td>Display Sample Rate</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
<tr><td>Value</td><td></td><td>Sample Rate.
0x00 - Unsupported
0x01 - 32KHz
0x02 - 44.1KHz
0x03 - 48KHz
0x04 - 88.2KHz
0x05 - 96KHz
0x06 - 176.4KHz
0x07 - 192KHz</td></tr>
</table></td></tr>
<tr><td>Notes</td><td></td></tr>
</table>

Chromecast App Name

<table>
<tr><td>Message</td><td>Chromecast App Name</td></tr>
<tr><td>Direction</td><td>Avatar -> Device</td></tr>
<tr><td></td><td>This message allows the Avatar to notify the Device that the Chromecast App Name has changed
<table>
<tr><td colspan="3">Request : Avatar->Device</td></tr>
<tr><td>Message</td><td>0x0004</td><td>State Change</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0x0304</td><td>Chromecast App Name</td></tr>
<tr><td>Encoding</td><td>0xC5</td><td>BIN16</td></tr>
</table></td></tr>
</table>

	Value		This will contain the name of the ChromeCast provider.
Notes			

Album Art URL

Message	Album Art URL		
Direction	Avatar -> Device		
	This message allows the Avatar to notify the Device of the current Album Art URL		
	Request : Avatar->Device		
	Message	0x0004	State Change
	Count	1	
	State	0x0307	Album Art URL
	Encoding	0xC5	BIN16
	Value		This will contain the URL of the current Album Art as an SHA256 Hash
Notes			

Friendly Name

Message	Friendly Name		
Direction	Avatar -> Device		
	This message allows the Avatar to notify the Device when the Friendly Name is changed		
	Request : Avatar->Device		
	Message	0x0004	State Change
	Count	1	
	State	0x0311	Friendly Name
	Encoding	0xC5	BIN16
	Value		This will contain the Friendly Name assigned to the Avatar
Notes	Anam have requested that this be limited to a maximum of 30bytes		

MQA State

<table>
<tr><td>Message</td><td>MQA State</td></tr>
<tr><td>Direction</td><td>Avatar -> Device</td></tr>
<tr><td></td><td>This message allows the Avatar to notify the Device when the MQA playback state is changed
<table>
<tr><th colspan="3">Request : Avatar->Device</th></tr>
<tr><td>Message</td><td>0x0004</td><td>State Change</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0x0312</td><td>MQA State</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
<tr><td>Value</td><td></td><td>This will contain current MQA state
0x00 No MQA
0x01 MQA
0x02 MQA Studio
0x03 MQA ORFS</td></tr>
</table></td></tr>
<tr><td>Notes</td><td>The available values are based on the MQA Playback UI and Marketing Guidelines v14 document</td></tr>
</table>

Region

<table>
<tr><td>Message</td><td>Region</td></tr>
<tr><td>Direction</td><td>Avatar -> Device</td></tr>
<tr><td></td><td>This message allows the Avatar to notify the device that the WiFi Region has changed.
<table>
<tr><th colspan="3">Request : Avatar->Device</th></tr>
<tr><td>Message</td><td>0x0002</td><td>Get States</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0x0313</td><td>Region</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
<tr><td>Value</td><td></td><td>Region
0x00 - Unsupported
0x01 - US
0x02 - CA
0x03 - TW</td></tr>
</table></td></tr>
</table>

		• 0x04 - SG • 0x05 - DE • 0x06 – KR • 0x07 – VN • 0x08 – MY • 0x09 – PH • 0x0A – NZ • 0x0B – AU • 0x0C – AE • 0x0D – ZA • 0x0E – HR • 0x0F – JP • 0x10 – RU • 0x11 – IN • 0x12 – CN • 0x13 – ID • 0x14 – IL • 0x15 – BR

Response : Avatar->Device		
Message	0x0004 0x8002	State Change Get States Response
Count	1	
State	0x0313	Region
Encoding	0xCC	UINT8
Value		**Region** • 0x00 - Unsupported • 0x01 - US • 0x02 - CA • 0x03 - TW • 0x04 - SG • 0x05 - DE • 0x06 – KR • 0x07 – VN • 0x08 – MY • 0x09 – PH • 0x0A – NZ • 0x0B – AU • 0x0C – AE • 0x0D – ZA • 0x0E – HR • 0x0F – JP

				• 0x10 – RU • 0x11 – IN • 0x12 – CN • 0x13 – ID • 0x14 – IL • 0x15 – BR
Notes	The Avatar Region message may be sent either in response to a Region Get States request, or as a State Change.			

Album Art

Message	Album Art		
Direction	Device -> Avatar		
	This message allows the device to request the current album art from the Avatar. The Avatar will download the album art from the appropriate URL, convert it to a fixed resolution dependent on the device model, and then transfer it as a JPEG using the Album Art message.		

Request : Device->Avatar		
Message	0x0001	Set States
Count	1	
State	0x0308	Album Art
Encoding	0xC5	BIN16
Value		This will contain the URL of the requested Album Art

Request : Avatar->Device		
Message	0x0001	Set States
Count	1	
State	0x0307	Album Art URL
Encoding	0xC5	BIN16
Value		This will contain the URL of the current Album Art as an SHA256 Hash
State	0x0308	Album Art
Encoding	0xC5	BIN16

	Value		This will contain the album art as jpeg data.
Notes	Note the message containing the Album Art data will only be initiated once the image has been downloaded, and is a new Set States message rather than a Get States Response This may require more than one Set States message (with incrementing Packet Id's) to be able to transfer all the Album Art data. We will be able to identify when all the data has been sent by looking at the packets control flags (FEND = 1)		

Serial Number / MAC Address

Message	Get Serial Number /MAC Address		
Direction	Avatar -> Device		
	This message allows the device to request both the Serial Number and MAC Address		
	Request : Device->Avatar		
	Message	0x0002	Get States
	Count	2	
	State	0x0314	Set Serial Number
	Encoding	0xC5	BIN16
	Value		This will contain the Serial Number
	State	0x0315	Set MAC Address
	Encoding	0xC5	BIN16
	Value		This will contain the MAC address in string format e.g. 01:23:45:67:89:ab
Notes	It is expected that the two States are included in the same message.		

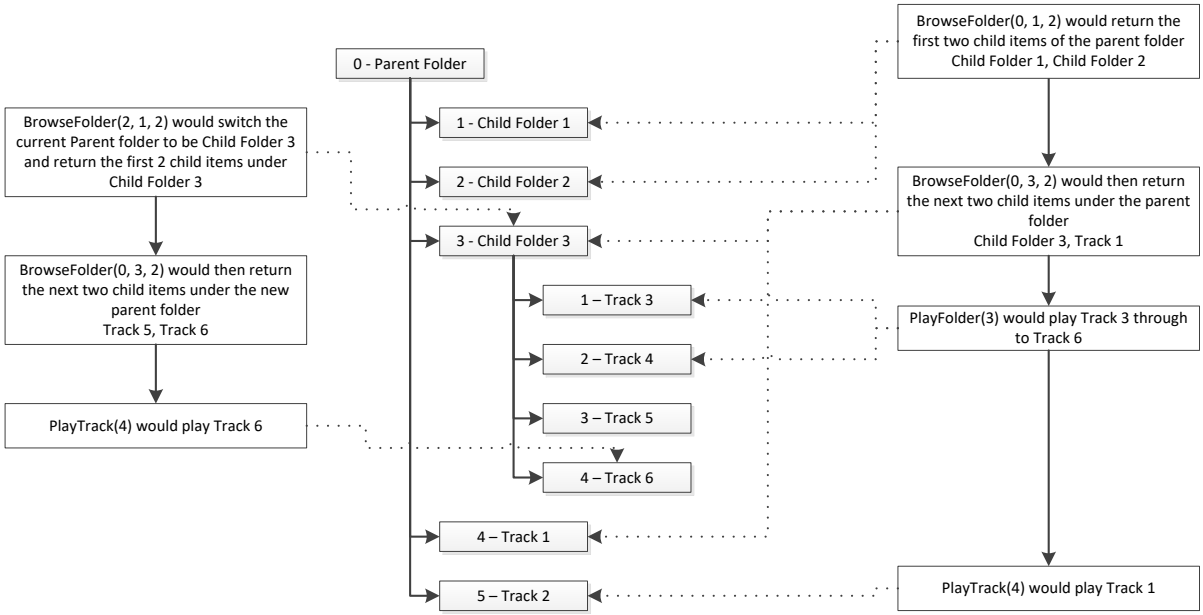
Questions / Notes
1. Matjaz couldn't find any spec limiting the Chromecast App Name 2. There is no need to include MAC Addresses or Hostname here as they are passed in the Avatar System Info message during handshaking.

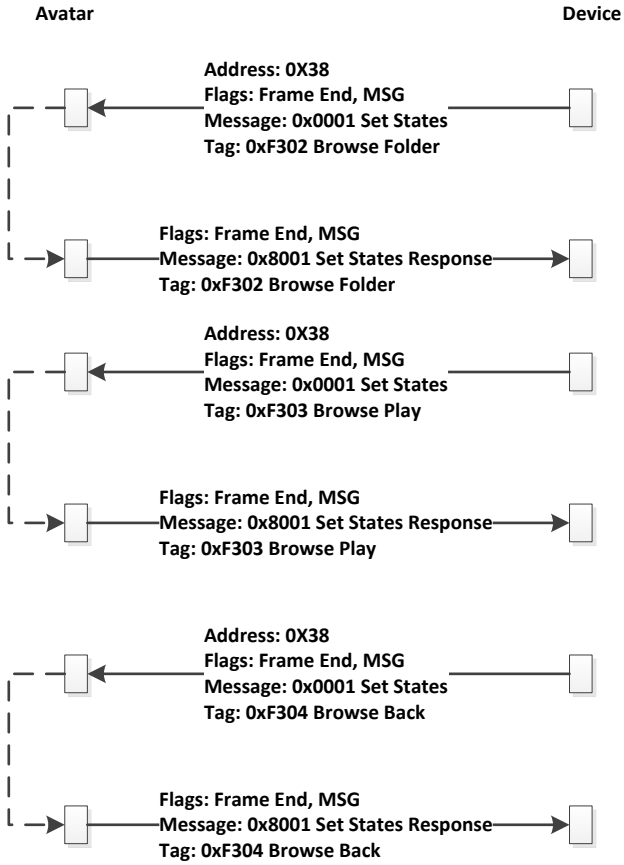
Browsing

Browsing will give the user the ability to navigate a hierarchy of folders and tracks.

The approach to navigation will be as follows:

- Standard browsing starts at the root folder.
- Each item (Folder/Track) contained in a Folder will be given a unique ascending identifier, starting at 1 such that the identifier of the item preceding and following a given item can be determined by decrementing/incrementing the current items identifier.
- The identifier 0 will always be used to refer to the current Folder.





Browse Folder

<table>
<tr><td>Message</td><td>Browse Folder</td></tr>
<tr><td>Direction</td><td>Device -> Avatar</td></tr>
<tr><td></td><td>This message is used to indicate to the Avatar that the device wishes to either:
Return items from the current folder
Navigate to a child folder owned by the current folder
When navigating to a child folder the child folder will become the current folder.

<table>
<tr><td colspan="3">Request : Device->Avatar</td></tr>
<tr><td>Message</td><td>0x0001</td><td>Set States</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0xF302</td><td>Browse Folder</td></tr>
<tr><td>Encoding</td><td>0xC5</td><td>BIN16</td></tr>
<tr><td>Value</td><td></td><td>Byte</td></tr>
</table></td></tr>
</table>

		1-2	The Identifier of the Folder to browse. 0 – Indicates the current folder >0 – Indicates the identifier of a child folder in the current folder.
		3-4	The identifier of the first item in the list of items to be returned in the response
		5-6	The maximum number of items to be returned.

Response : Avatar->Device		
Message	0x8001	Set States Response
Count	1	
State	0xF302	Browse Folder
Encoding	0xC5	BIN16
Value		**Byte**

1-2	Actual number of items returned
For each item returned	
N-N+1	Item Identifier
N+2	Item Type 0x00 Folder 0x01 Track
N+3-N+43	The Name of the item

Notes	Where the call to Browse Folder requests more items that are available, all available items matching the criteria will be returned. If the Folder identifier does not reference a folder, (i.e. it is a Track) then 0 items will be returned.

Browse Play

<table>
<tr><td>Message</td><td>Browse Play</td></tr>
<tr><td>Direction</td><td>Device -> Avatar</td></tr>
<tr><td></td><td>This message is used to indicate to the Avatar that play the Track, or the contents of the Folder associated with the identifier from the current folder

<table>
<tr><td colspan="3">Request : Device->Avatar</td></tr>
<tr><td>Message</td><td>0x0001</td><td>Set States</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0xF303</td><td>Browse Play</td></tr>
<tr><td>Encoding</td><td>0xCD</td><td>UINT16</td></tr>
<tr><td>Value</td><td></td><td>The identifier of the Track or Folder to be played in the current folder</td></tr>
</table>
</td></tr>
<tr><td>Notes</td><td></td></tr>
</table>

Browse Back

<table>
<tr><td>Message</td><td>Browse Back</td></tr>
<tr><td>Direction</td><td>Device -> Avatar</td></tr>
<tr><td></td><td>This message is used to indicate to the Avatar that we wish to browse back up the hierarchy.

It will include the following information:

• Where to browse up to e.g.
  ○ Up one level
  ○ To Root

Calling this will make the selected folder current. The Browse Page message can then be called to get items, or Select Item to play the tracks in the folder..

The response will contain the ID of the selected folder.

<table>
<tr><td colspan="3">Request : Device->Avatar</td></tr>
<tr><td>Message</td><td>0x0001</td><td>Set States</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0xF304</td><td>Browse Back</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
</table>
</td></tr>
</table>

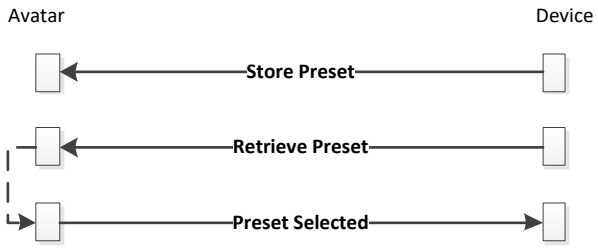
<table>
<tr><td></td><td>
<table>
<tr><td>Value</td><td></td><td>Level
0x00 Root folder
0x01 Up one folder</td></tr>
</table>
<table>
<tr><th colspan="3">Response : Avatar->Device</th></tr>
<tr><td>Message</td><td>0x8001</td><td>Set States Response</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0xF304</td><td>Browse Back</td></tr>
<tr><td>Encoding</td><td>0xC5</td><td>BIN16</td></tr>
<tr><td>Value</td><td></td><td>The ID and name of the selected folder.
Byte
<table>
<tr><td>1-40</td><td>The Name of the new current Folder</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Notes</td><td>If browse back is called on the root folder, the current folder will remain as the root folder.

There should be no need to return the identifier for the Folder as this call makes it the current folder which always has the identifier = 0</td></tr>
</table>

Questions / Notes
• Our current approach to Browsing is that we do not need to handle dynamic changes to the contents of a level while that level is being browsed and that the state of the level at the time it was entered will be sufficient. • Peters response from 21/11/2018 says that their plan is to only refresh when navigation to a child level. Parent level data will be cached. This should not be a problem? • The Avatar will keep track of the current folder, and its parent. • This approach to identifiers means that the device will need to be able to predict the order of identifiers in list. That leads to the conclusion that the identifier should be a simple integer based on an items ordinal position in the list. • We will need to define in which order folders/tracks are added to the list i.e. alphabetical, or folders first followed by tracks.

Pre-sets

Pre-sets allow the user to store settings that can easily be recalled for playback. At this point we are not clear on the types of setting that can be recalled e.g. Album/Artist/Playlist.



Store Pre-set

Message	Store Pre-set
Direction	Device -> Avatar

	This message will indicate to the Avatar that its current state should be saved into a pre-set.
	The message will contain:
	• A pre-set identifier (assuming there is more than one pre-set)
	• Some data to identify what is to be stored in the pre-set (see below)
	  ○ Active Input
	  ○ Artist/Album/Track/Playlist

Notes	We need clarification on what can be stored as a pre-set.

Retrieve Pre-set

Message	Retrieve Pre-set
Direction	Device -> Avatar
	This message will indicate to the Avatar to load the specified pre-set. The message will contain: • A pre-set identifier (assuming there is more than one pre-set)
Notes	

Pre-set Selected

Message	Pre-set Selected
Direction	Avatar -> Device
	This message will indicate to the device that the Avatar has loaded a pre-set. The message will contain: • A pre-set identifier (assuming there is more than one pre-set)
Notes	

Questions / Notes

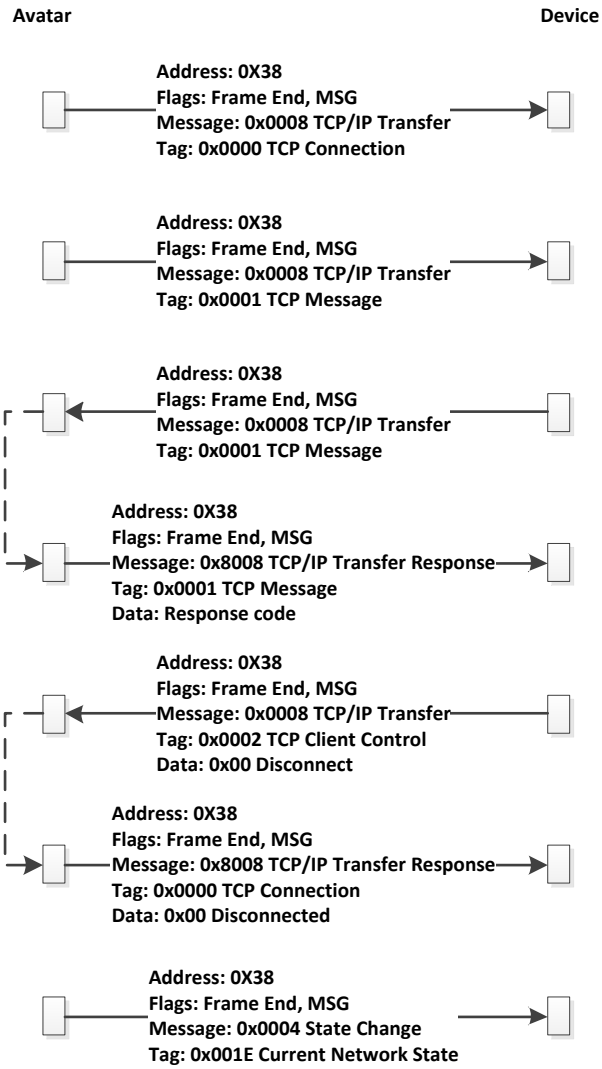
- We need to identify what state can be stored as a pre-set. If there is a choice e.g. album, or track then we need to be able to identify which? This may affect the number of messages required.
- We need to decide how many pre-sets we want to make available
- Do we want to be able to name pre-sets?
- The Citation modules do not support pre-sets

Network

The ARCAM app and RS232 command line specification allows the device to be controlled over TCP/IP, however the device will not have its own dedicated network port and will rely on the Avatar tunnelling any data intended for the device through to it.

The tunnelling must be able to handle the following scenarios:

- Multiple client connections / disconnections (probably 4)
- Passing external data received on port 50000 to the device.
- Sending data from the device to a specific client connection. This may occur as the result of some previously received data or may be unsolicited.



TCP Client Connection

<table>
<tr><td>Message</td><td>TCP Client Connected</td></tr>
<tr><td>Direction</td><td>Avatar -> Device</td></tr>
<tr><td></td><td>Sent when a new TCP Client connection is established or disconnected.

The message will include a unique identifier per client connection that can be used to identify the client sufficient for unsolicited messages to be generated to that client.

<table>
<tr><td colspan="3">Request : Avatar->Device</td></tr>
<tr><td>Message</td><td>0x0008</td><td>TCP/IP Transfer</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0x0000</td><td>TCP Client Connection</td></tr>
<tr><td>Encoding</td><td>0xCD</td><td>UINT16</td></tr>
</table>
</td></tr>
</table>

	Value		**Byte** | 1 | Client Identifier 2 | **Connection** • 0x00 Disconnected • 0x01 Connected
Notes	To save on the number of values returned in the message the 2 UINT8 data items are concatenated into a single UINT16. The byte order of the value returned will be little-endian with Byte 1 (Client identifier) coming first and Byte 2 (Connection) second.		

TCP Client Message

Message	TCP Message		
Direction	Avatar -> Device		
	Sent when a TCP Message (or data) is sent or received. It is expected that some messages will require multiple packets to be sent to pass the data. The message will contain: • The unique identifier from the Client Connection message • Message data		
	Request : Avatar->Device, Device->Avatar		
	Message	0x0008	TCP/IP Transfer
	Count	1	
	State	0x0001	TCP Client Message
	Encoding	0xC5	BIN16
	Value		**Byte** | 1-2 | Data Length – The total length of the data to be transferred (Only included in the first packet) 3 | Client Identifier (Only included in the first packet) 4-N | Message data.

<table>
<tr><td></td><td>
<table>
<tr><td colspan="3">Response : Avatar->Device</td></tr>
<tr><td>Message</td><td>0x8008</td><td>TCP/IP Transfer Response</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0x0001</td><td>TCP Client Message</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
<tr><td>Value</td><td></td><td>An enum representing the success/failure of the transfer.

Transfer State

• 0x00 Successful Transfer
• 0x01 Invalid Message
○ The received HostLink message is malformed
• 0x02 Invalid Connection
○ A connection with the given ID doesn’t exist
• 0x03 Transfer Failed
○ Writing to the TCP socket failed
• 0x04 Invalid Command
○ Unknown control ID</td></tr>
</table>
</td></tr>
<tr><td>Notes</td><td>Originally, the message included an Offset though this was removed at the request of Stream.

The Data Length and Client Id will only be included in the first packet, subsequent packets will only contain the data.

We are now assuming that there will be a fixed maximum message size of 512bytes, and that the Offset in a particular packet can be calculated using the packet id e.g.

Offset = packet_id * 512

We will be able to identify when all the data has been sent by looking at the packets control flags (FEND = 1).

In reality, it is unlikely that any of the TCP data received will exceed 256 bytes though this will need review when we start sending Artist/Album/Track info out from the host.</td></tr>
</table>

TCP Client Control

Message	TCP Client Control
Direction	Device -> Avatar

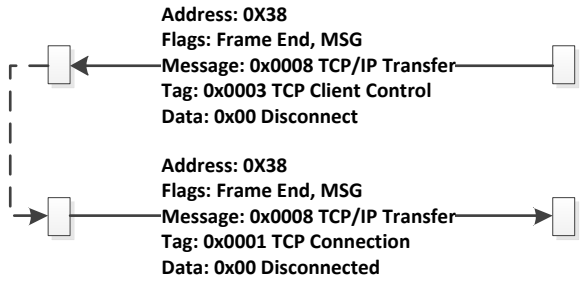
Allows the Device to control a TCP connection.

Request : Device->Avatar		
Message	0x0008	TCP/IP Transfer
Count	1	
State	0x0002	TCP Client Control
Encoding	0xCD	UINT16
Value		**Byte** | 1 | Client Identifier 2 | **Control** • 0x00 Disconnect

Response : Avatar->Device		
Message	0x8008	TCP/IP Transfer Response
Count	1	
State	0x0002	TCP Client Control
Encoding	0xCC	UINT8

Value		An enum representing the success/failure of the command.
		Transfer State
		• 0x00 Successful Transfer
		• 0x01 Invalid Message
		  ○ The received HostLink message is malformed
		• 0x04 Invalid Command
		  ○ Unknown control ID

Notes	To save on the number of values returned in the message the 2 UINT8 data items are concatenated into a single UINT16. The byte order of the value returned will be little-endian with Byte 1 (Client identifier) coming first and Byte 2 (Control) second. In version 1 only the Disconnect option is available The Avatar will respond to a Disconnect message with a TCP Client Connection (0x00 Disconnected message)

	20190425 - To Check – Stream currently responding with a TCP Client Control message – not a problem but we will need to update this section if we stick with that. 

Current Network State

Message	Current Network State		
	The Avatar indicates the state of the Network connection.		
	Request : Avatar->Device		
	Message	0x0004	State Change
	Count	1	
	State	0x001E	Network State • 0 – Not Connected • 1 – Connected • 2 – Connecting • 3 - Error
Notes	This message is a standard Hostlink message that is used to identify when the user has “Forgotten” the Avatar network settings (where a 3 – Error state will be returned).		

Questions / Notes

- A unique identifier is required for each client connection, which will be a simple integer.
- There is no need for UDP messaging as it is only required for the Control4/Crestron identification. In these scenarios, the device will only send messages that trigger UDP messages in the Avatar rather than having to sending actual UDP messages that require UDP IP Addresses/Ports.
- A new Message ID pair will be created specifically for TCP/IP Transfers

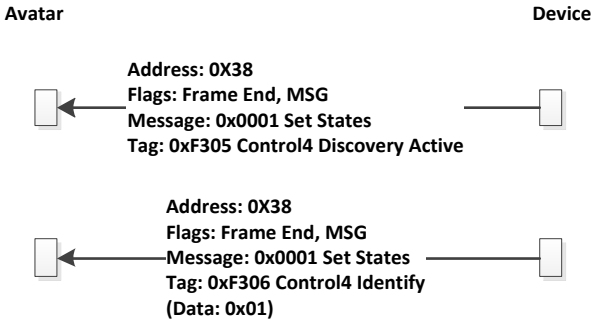
Value	Message Type	Comment	Description
0x0008	Tagged Value List	TCP/IP Transfer	Tag = state id, value = new state

0x8008	Tagged Value List	TCP/IP Transfer Response	Tag = original state id, value = Nil if unsupported, uint8(error enum) otherwise.

- There is no need for the Device to send a Transfer Response message (Matjaz email 11/04/2019)

Control4

Control4 network identification will be handled by the Avatar, however there is a requirement for the user to be able to trigger a Network identify message which is a manually generated unsolicited message generated by the device when connecting to a Control 4 Controller.



Control4 Discovery Active

Message	Control4 Discovery Active
Direction	Device -> Avatar
	Sent by the Device to inform the Avatar whether the Control4 Discovery should be active or not. When set to Inactive the Avatar should not respond to Control4 Discovery Requests

Request : Device->Avatar		
Message	0x0001	Set States
Count	1	
State	0xF305	Control4 Discovery Active
Encoding	0xCC	UINT8
Value		Control 4 Discovery Active Options: • 0x00 - Control4 Discovery Inactive • 0x01 - Control4 Discovery Active

Notes	

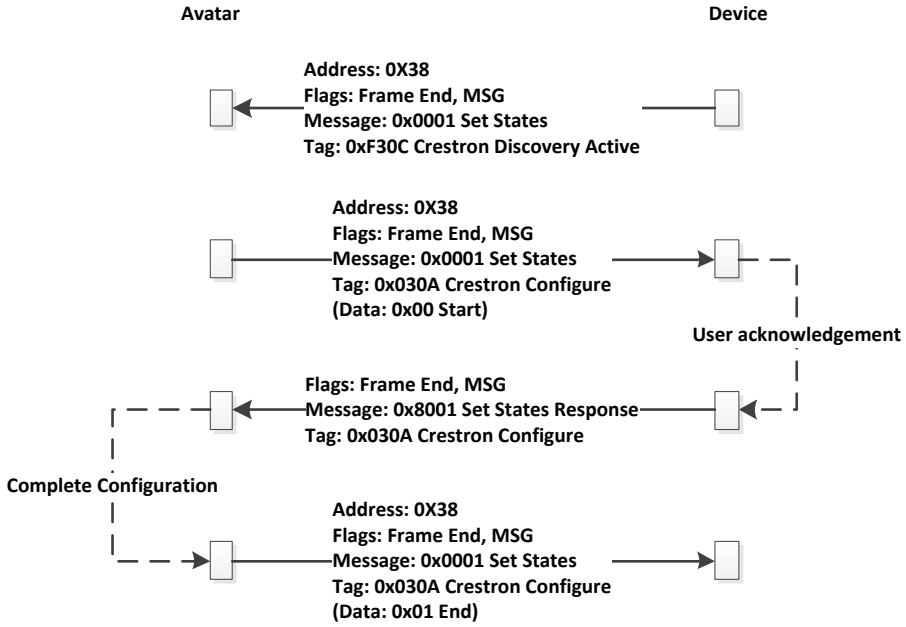
Control4 Identify

<table>
<tr><td>Message</td><td>Control4 Identify</td></tr>
<tr><td>Direction</td><td>Device -> Avatar</td></tr>
<tr><td></td><td>Sent by the Device in response to a user action. This informs the avatar that a Control4 Notify Identify message should be sent.
There is no need for a payload in this message
There is no need for the Avatar to respond (other than the ACK)
<table>
<tr><th colspan="3">Request : Device->Avatar</th></tr>
<tr><td>Message</td><td>0x0001</td><td>Set States</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0xF306</td><td>Control4 Identify</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
<tr><td>Value</td><td></td><td>0x01 (an arbitrary value)</td></tr>
</table></td></tr>
<tr><td>Notes</td><td></td></tr>
</table>

Questions / Notes
• Not all the devices supported will have the ability to trigger Notify Identify messages. In these cases this functionality will be remain in this API but will not be used by the Device.

Crestron

Crestron network identification will be handled by the Avatar, however as part of the process the device is required to prompt the user to start the configuration process by pressing a button and relay to the Avatar that the button has been pressed.



Crestron Discovery Active

Message	Crestron Discovery Active
Direction	Device -> Avatar
	Sent by the Device to inform the Avatar whether the Crestron Discovery should be active or not. When set to Inactive the Avatar should not respond to Crestron Discovery Requests

Request : Device->Avatar

Message	0x0001	Set States
Count	1	
State	0xF30C	Crestron Discovery Active
Encoding	0xCC	UINT8
Value		Crestron Discovery Active Options: • 0x00 - Crestron Discovery Inactive • 0x01 - Crestron Discovery Active

Notes	

Crestron Configure

Message	Crestron Configure
Direction	Avatar -> Device

<table>
<tr>
<td></td>
<td>
Sent by the Avatar to the Device when the avatar receives the Start Light N Poll message from the Crestron controller, and when the avatar receives the Light N Poll Stop message.

The response will only be sent for the Start Configuration (0x00) and will be sent once the user has acknowledged the Crestron Configuration prompt. This will trigger the Button Pressed Creston message to be sent by the Avatar.

<table>
<tr><td colspan="3">Request : Avatar->Device</td></tr>
<tr><td>Message</td><td>0x0001</td><td>Set States</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0x030A</td><td>Crestron Configure</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
<tr><td>Value</td><td></td><td>• 0x00 Start Configuration
• 0x01 Stop Configuration</td></tr>
</table>

<table>
<tr><td colspan="3">Response : Device->Avatar</td></tr>
<tr><td>Message</td><td>0x8001</td><td>Set States Response</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0x030A</td><td>Crestron Configure</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
<tr><td>Value</td><td></td><td>0x01 (Arbitrary value)</td></tr>
</table>
</td>
</tr>
<tr>
<td>Notes</td>
<td>
There will conceivably be a gap between the Avatar sending the Set States message and the Device responding as it requires user action. Is this approach (Set States / Response) OK for this type of scenario?

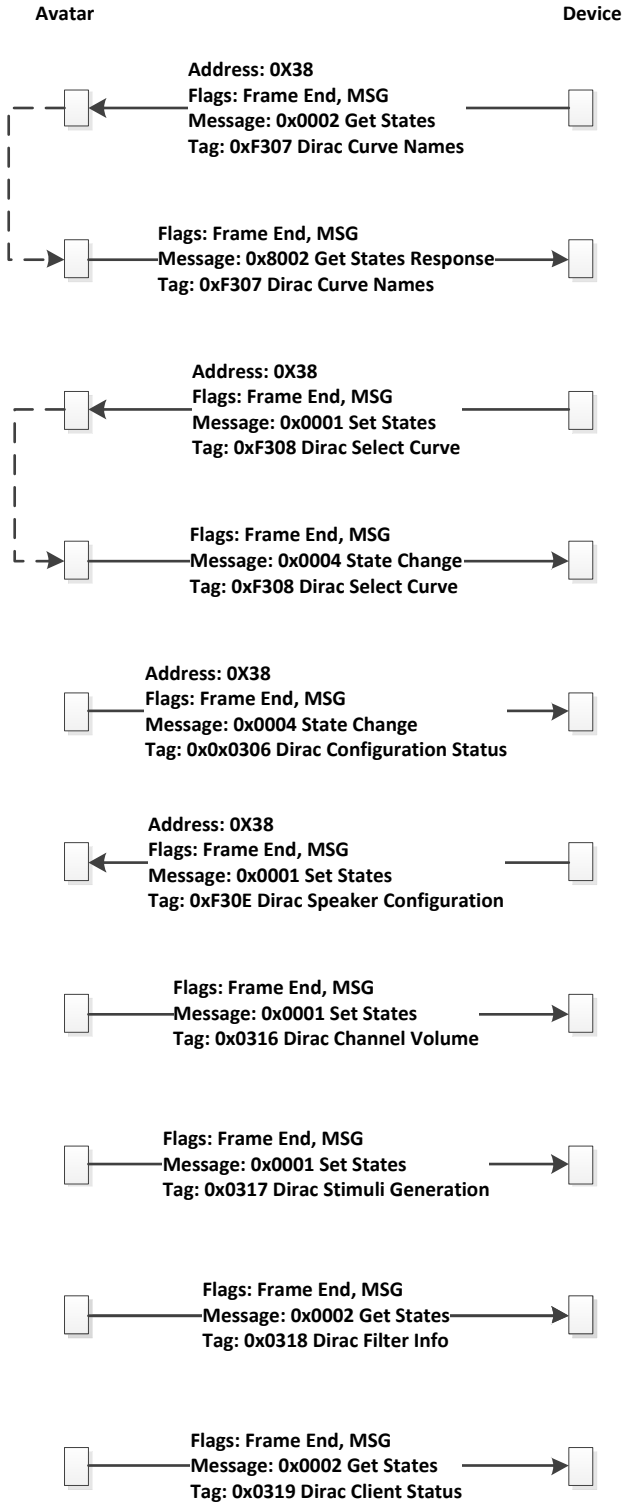
This has been implemented as a Set States message pair should it be a Key event instead?

A Response will only be sent for the Start Configuration message once the user presses the configure button.
</td>
</tr>
</table>

Questions / Notes

Dirac

The Avatar will implement Dirac room correction. It will be possible to display, and select the currently selected EQ configuration from the device, and to turn Dirac on and off.



Dirac Curve Names

Message	Get Dirac Curve Names
Direction	Device -> Avatar
	Sent by the Device to request the DIRAC curve names. This will trigger the Avatar Dirac Curve Names message.

Request : Device->Avatar

Message	0x0002	Get States
Count	1	
State	0xF307	Dirac Curve Names
Encoding	0xCC	UINT8
Value		0x01 (Arbitrary value)

Response : Avatar->Device

Message	0x8002	Get States Response
Count	1	
State	0xF307	Dirac Curve Names
Encoding	0xC5	BIN16
Value		

Byte	
1-2	Data Length
3	Number of Curves
For each curve	
4+(N*?)	Dirac Curve Identifier (UINT8) This is the index of the curve in DIRAC and may not be sequential in relation to the ordinal position of the curve in the list. The only restriction is that the first curve passed in the list represents No Curve.
4+(N*?) + 1 - ?	Dirac Curve Name (fixed length)

Notes	This may be triggered after handshaking, during the Device configuration phase, or later when the curve names are about to be displayed (TBD) The values passed for the Dirac Curve name will be a fixed 40 byte length field padded with zeros (Note if the Curve name is > 40 bytes then a null terminator is not required). This is to enable the host code to parse the contents of the BIN16 field.

Dirac Select Curve

<table>
<tr><td>Message</td><td>Dirac Select Curve</td></tr>
<tr><td>Direction</td><td>Device -> Avatar</td></tr>
<tr><td></td><td>Instructs the Avatar to select a specific Dirac Curve. The Avatar does not need to respond directly to the Set States message but will raise a State Changed notification.
<table>
<tr><th colspan="3">Request : Device->Avatar</th></tr>
<tr><td>Message</td><td>0x0001</td><td>Set States</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0xF308</td><td>Dirac Select Curve</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
<tr><td>Value</td><td></td><td>Dirac Curve Identifier (obtained from Dirac Get Curves (0xF307)

This is the index of the curve in DIRAC and may not be sequential in relation to the ordinal position of the curve in the list.

The only restriction is that the first curve passed in the list represents No Curve.</td></tr>
</table>
<table>
<tr><th colspan="3">Response : Avatar->Device</th></tr>
<tr><td>Message</td><td>0x0004</td><td>State Change</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0xF308</td><td>Dirac Select Curve</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
<tr><td>Value</td><td></td><td>Dirac Curve Identifier (obtained from Dirac Get Curves (0xF307)

This is the index of the curve in DIRAC and may not be sequential in relation to the ordinal position of the curve in the list.

The only restriction is that the first curve passed in the list represents No Curve.</td></tr>
</table>
This message will also be used to switch off Dirac by specifying No Curve (0x00)</td></tr>
<tr><td>Notes</td><td>This will be the message used to Enable/Disable Dirac (0xF308).</td></tr>
</table>

	The Avatar will respond to a Dirac Select Curve message and may generate unsolicited Dirac Select Curve messages. In the initial implementation the device will ignore any State Change message sent by the Avatar. The Avatar should not select a curve when it is saved during calibration. Is it OK to adopt the approach of using a Set States message to instruct the avatar to do something then a State Change message to pick up the change?

Dirac Calibration Status

Message	Dirac Calibration Status
Direction	Avatar -> Device

Informs the Device that Dirac is being calibrated.

Request : Device->Avatar		
Message	0x0004	State Change
Count	2	
State	0x0306	Dirac Calibration Status
Encoding	0xCC	UINT8
Value		**Calibration Status** • 0X00 – Calibration Started • 0X01 – Calibration Stopped • 0x02 – Calibration Updated
State	0xF307	Dirac Curve Names
Encoding	0xC5	BIN16
Value		

Byte	
1-2	Data Length
3	Number of Curves
For each curve	
4+(N*?)	Dirac Curve Identifier (UINT8) This is the index of the curve in DIRAC and may not be sequential in relation to the ordinal position of the curve in the list.

				The only restriction is that the first curve passed in the list represents No Curve.
			4+(N*?) + 1 - ?	Dirac Curve Name
Notes	When receiving the Calibration Started message the device will change its display to reflect that calibration is underway. This will be triggered by the Avatar playing either the Test tone or Calibration tone. The Dirac Curve Names state will return the Id/Name of all the curves. The device will be responsible for working out where changes have occurred.			

Dirac Speaker Configuration

Message	Dirac Speaker Configuration		
Direction	Device -> Avatar		
	Upon receiving a new speaker configuration, old filters are deleted and Dirac IO is restarted using the new config		
	Request : Device->Avatar		
	Message	0x0001	Set States
	Count	1	
	State	0xF30E	Dirac Speaker Configuration
	Encoding	0xCC	UINT8
	Value		Number of Speakers in the Configuration
	For each speaker		
	Encoding	0xCC	UINT8
	Value		Speaker type: • 0x00 - Subwoofer • 0x01 – Small 40Hz • 0x02 – Small 50Hz • 0x03 – Small 60Hz • 0x04 – Small 70Hz • 0x05 – Small 80Hz • 0x06 – Small 90Hz • 0x07 – Small 100Hz • 0x08 – Small 110Hz • 0x09 – Small 120Hz • 0x0A – Small 150Hz • 0x0B – Small 160Hz

			• 0x0C – Small 170Hz • 0x0D – Small 180Hz • 0x0E – Small 190Hz • 0x0F – Small 200Hz
Notes	Confirm Speaker types enumeration is complete		

Dirac Channel Volume

Message	Dirac Channel Volume		
Direction	Avatar -> Device		
	When user changes per-channel slider gain, this message is sent to the DSP, which should apply it to the stimuli immediately (if playing)		
	Request : Avatar -> Device		
	Message	0x0001	Set States
	Count	1	
	State	0x0316	Dirac Channel Volume
	Encoding	0xCC	UINT8
	Value		Channel Index
	Encoding	0xCF	UINT64 (Used to hold a 64bit double)
	Value		The gain that should be applied to the Channel.
Notes	Confirm that a 64bit integer field is OK for storing a double?		

Dirac Stimuli Generation

Message	Dirac Stimuli Generation
Direction	Avatar -> Device
	When a stimuli is triggered/stopped/ongoing, the DSP should generate the stimuli from given parameters and start/stop playback (if necessary). The volume is controlled by the Dirac Channel Volume and Volume messages

Request : Avatar -> Device		
Message	0x0002 0x0004	Get States State Change
Count	1	
State	0x0317	Dirac Stimuli Generation
Encoding	0xCC	UINT8
Value		Playback Active: • 0x00 – False • 0x01 – True
Encoding	0xCC	UINT8
Value		Stimuli Type: • 0x00 – Pink Noise • 0x01 – Log Sweep
Depending on the Stimuli Type		
Pink Noise		
Encoding	0xCE	UINT32
Value		Sample Rate
Encoding	0xCC	UINT8
Value		Playback length (in seconds)
Log Sweep		
Encoding	0xCD	UINT16
Value		Start Frequency (Hz)
Encoding	0xCD	UINT16
Value		Stop Frequency (Hz)
Encoding	0xCC	UINT8
Value		Playback Length (Seconds)
Encoding	0xCC	UINT8
Value		Silence Length (Seconds)
Encoding	0xCC	UINT8
Value		Number of Periods
Encoding	0xCE	UINT32

	Value		Sample Rate
	Encoding	0XCD	UINT16
	Value		Fade Out Tail (Hz)
Notes			

Dirac Filter Info

Message	Dirac Filter Info		
Direction	Avatar -> Device		
	When a filter is enabled, disabled or changed, this message is sent to the DSP so that it loads the appropriate FIIR coefficients.		
	Request : Avatar -> Device		
	Message	0x0002 0x0004	Get States State Change
	Count	1	
	State	0x0318	Dirac Filter Info
	Encoding	0xCD	UINT16
	Value		• Byte1: Channel Index • Byte2: Data Index
Notes			

Dirac Client Status

Message	Dirac Client Status		
Direction	Avatar -> Device		
	The Avatar informs the Device that the Dirac PC Client is connected		
	Request : Avatar -> Device		
	Message	0x0002 0x0004	Get States State Change
	Count	1	
	State	0x0319	Dirac Client Status

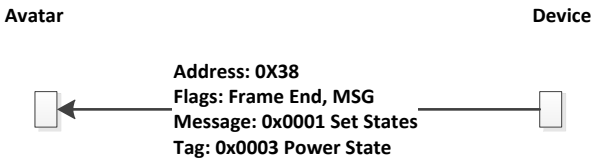
	Encoding	0xCC	UINT8
	Value		Connection Status: • 0x00 – Client Not Connected • 0x01 – Client Connected
Notes			

Questions / Notes

• The device will be responsible for switching Dirac On/Off during start-up using State ID 0xF308
• When saving a Dirac Curve the Avatar should not attempt to switch to the new curve.
• It should be possible to select Dirac curves on a per input basis. To do this the following approach will be taken on the device:
  ○ In Factory Settings state all inputs have Dirac Off
  ○ When the Device receives the first Calibration Updated (0x01) state it will set all inputs to use the first Dirac Curve in the list (ignoring No Curve)
  ○ When the Device receives a Calibration Updated (0x01) state it will enable the selection of any new Curves.

Power

The device will inform the Avatar when its Power state changes. This will include Power down and state transitions between Running and Sleep modes.



Power

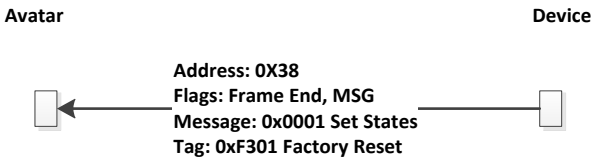
Message	Power		
Direction	Device -> Avatar		
	This message allows the device to inform the Avatar that its power state is changing.		
	Request : Device->Avatar		
	Message	0x0001	Set States

	Count	1	
	State	0x0003	Power State
	Encoding	0xCC	UINT8
	Value		**Power State** • 0x0A – Online • 0x14 – Standby • 0x1E – Offline
Notes	This will be identical to the existing Hostlink Set State message using the Power State ID (0x0003)		

Questions / Notes
For discussion between ARCAM and Stream Unlimited • We are not planning to inform the Avatar when the Device is powering down. However, when the device enters standby mode we will need to inform the Avatar. The Avatar will ideally be able to operate in a number of different standby states. ○ Sleep mode – The Device and Avatar are in as low-power mode as possible. There will be no network support. ○ Idle mode – The Avatar will be able to handle incoming network traffic but will not expose itself as a Castable device. ○ Matjaz response 29/03/2019 - At least for the Power states I can tell you right away that I don't anticipate that multiple standby states should be a problem. It's my understanding that we have 2 or 3 in the Citation codebase alone (called 'Idle', 'Standby', and 'LcdOff').

Factory Reset

The device will inform the Avatar that a Factory Reset is required.



Factory Reset

Message	Power
Direction	Device -> Avatar

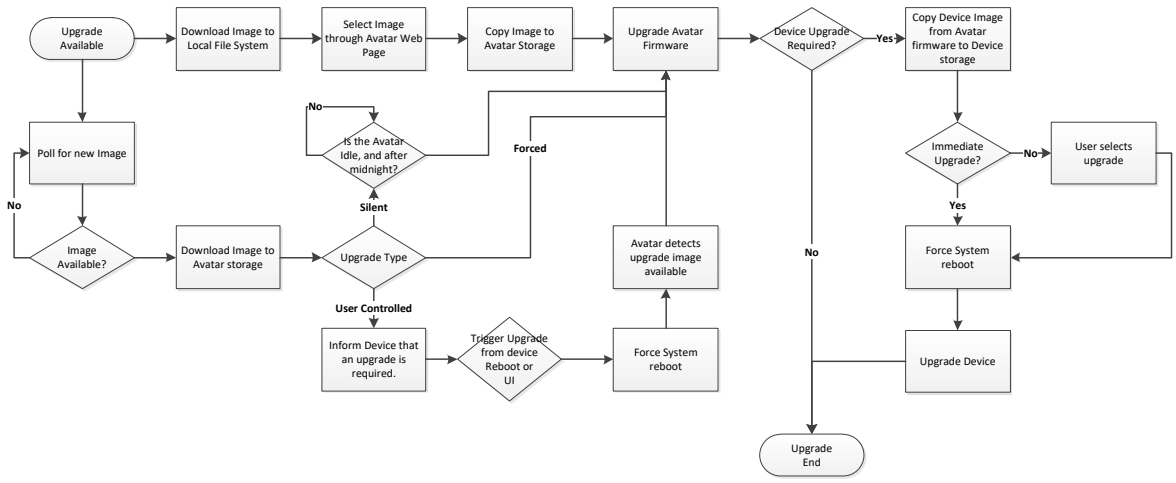
<table>
<tr><td></td><td>This message allows the device to inform the Avatar that a Factory Reset is required.
<table>
<tr><th colspan="3">Request : Device->Avatar</th></tr>
<tr><td>Message</td><td>0x0001</td><td>Set States</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0xF301</td><td>Factory Reset</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
<tr><td>Value</td><td></td><td>0x01 (Arbitrary value)</td></tr>
</table></td></tr>
<tr><td>Notes</td><td>This replaces the Citation Key Press Event 0x100B (Reset Long Press Events)</td></tr>
</table>

Questions / Notes

Upgrade

All firmware upgrades are controlled by the Avatar. If the Device requires an upgrade, the Device image is included in an Avatar upgrade image. The Avatar is upgraded, copying the Device image to the Avatars storage, which is then transferred to the Device for upgrade. The Avatar will inform the device when an upgrade is about to start.

Upgrade Scenarios

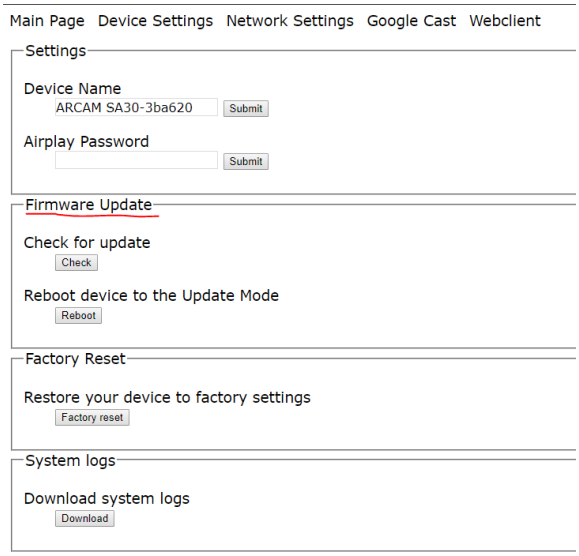


Local Network

In this scenario the user has identified that an upgrade image is available and has downloaded the image to the user's local network (PC/Hard Drive).

The user then chooses to install the new image by selecting the downloaded file through the Avatar Web Page.

The Avatar copies the Upload file contents into its own storage, and performs an upgrade.

Questions / Notes
Choosing the image for manual upgrade The new image can be installed through the Firmware Update option of the Avatar's Web Page. As this will require the user to identify the Upgrade Image file this can only be done this way, as the Device cannot browse the local networks files system(s).   

Automatic Upgrade

The Avatar checks the server every 4 hours for new Upgrade images.

If a new image is detected it is downloaded into the Avatars storage. Once downloaded the actual upgrade of the Avatar depends on the automatic upgrade option.

Forced Updates

If the Forced upgrade option is selected, then the Avatar will upgrade immediately. The Avatars firmware will be updated. If that includes a new version of the Device Firmware, the Device will be informed and the Device image transferred into the Devices storage, and the Device Upgrade triggered.

Silent Update

If the Silent upgrade option is selected, then the Avatar will wait until it is in an idle state, and that it is after midnight before performing the upgrade, following the same steps as a forced upgrade. If the Device is rebooted before then the upgrade will be performed immediately.

Questions / Notes

Upgrade Configuration

The Avatar currently supports the Forced and Silent upgrade types.

Silent Upgrades

Peter has said that the detection of the idle state can be extended to monitoring the Avatar external input. On the SA30 there is also the Analogue direct setting – in the scenario where Analogue Direct is on, the input to the Avatar will be disabled. As this is an edge case we are happy to put a note in the manual that this might happen.

Upgrades are usually done automatically before starting of StreamSDK process after firmware update. First, the Avatar needs to be upgraded. The device (host) binary is stored on the root file system of the Avatar (in /lib/firmware). After going from updater mode to standard mode and before starting StreamSDK processes a script using Host Link, and checks the version of the host (0xF001). If it is the same the StreamSDK process is started, if it is different, binary transfer is initialised.

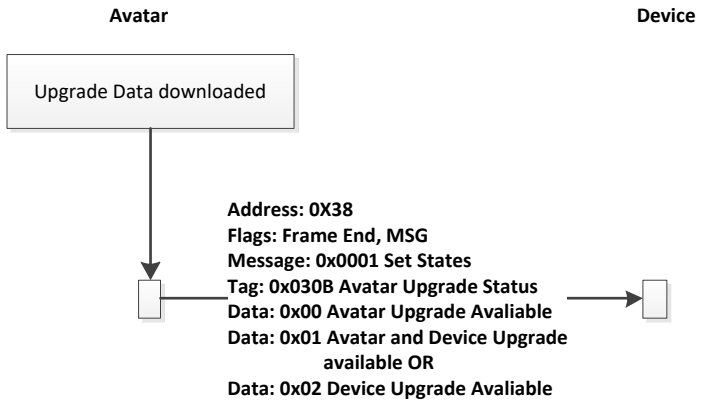
Device Upgrade Data

The structure of the Device upgrade data is device dependent. For the Solo Uno, SA30 and ST60 the layout of the Device Upgrade data will be as follows:

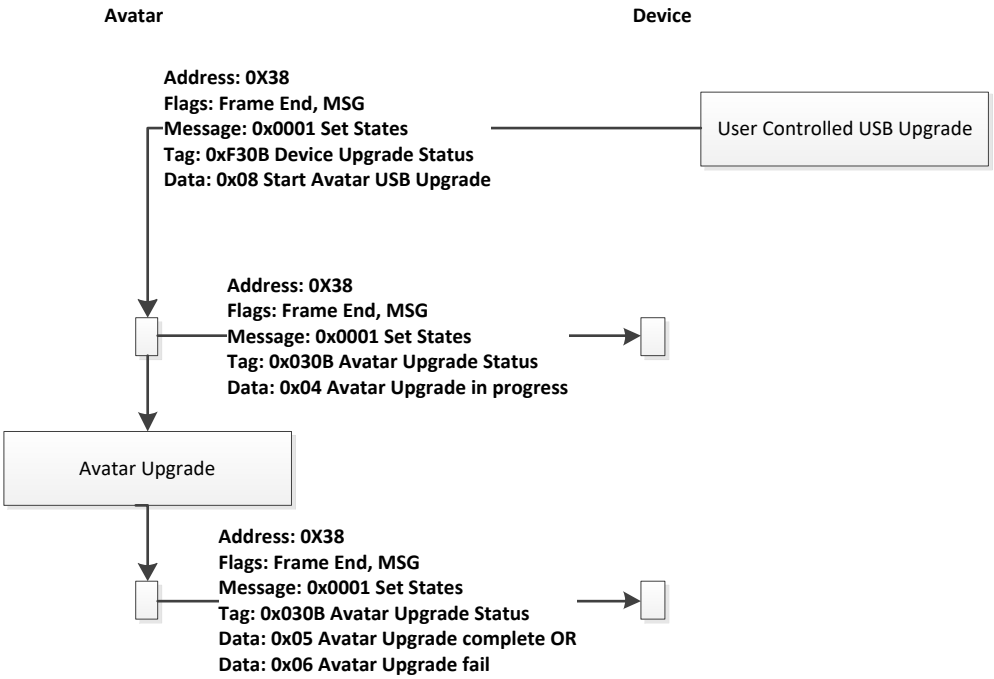
Byte	Encoding	Description
1	UINT8	Major Version
2	UINT8	Minor Version
3-6	UINT32	Image Data Size
7-22	BIN	MD5 Data Checksum (as a UINT8[16] array)
23-N	BIN	Image Data

Device Upgrade Process Status messages

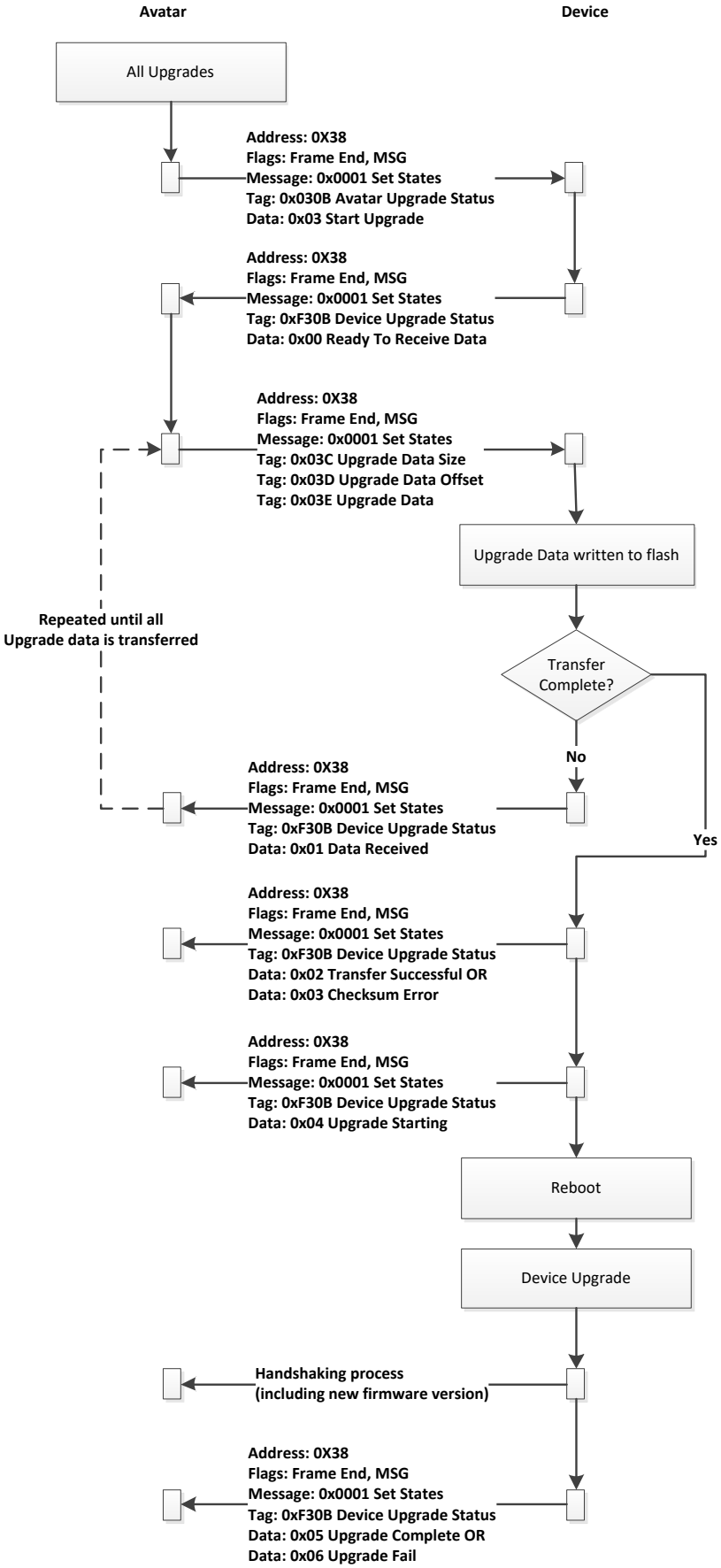
Upgrade Data Available



Avatar Firmware Upgrade



Device Firmware Data Transfer and Upgrade



Questions / Notes

Device upgrade complete message

The thing to note here is that the Device upgrade involves a reboot as it is the bootloader that performs the upgrade using the data transferred from the Avatar. Due to program size constraints, we do not intend to give the bootloader any Avatar communication capabilities, which means that we will not send an upgrade complete message until the handshaking process has completed between the Avatar and the Device on reboot. The handshaking process includes the Device notifying the Avatar of its firmware version, which will reflect the upgraded state of the Device prior to the Device informing the Avatar that it has completed its upgrade. If for any reason the upgrade fails the version reported during handshaking will be the old version, which could potentially cause a trigger of the whole Device upgrade process again unless the Avatar takes into account that the Device is still nominally in an upgrade until it send the Upgrade complete message.

Avatar Upgrade Option

<table>
<tr><td>Message</td><td>Upgrade Option</td></tr>
<tr><td>Direction</td><td>Device -> Avatar</td></tr>
<tr><td></td><td>This message is used to set the Upgrade mechanism. Initially it will be used to define whether OTA upgrades can be performed.

<table>
<tr><td colspan="3">Request : Device->Avatar</td></tr>
<tr><td>Message</td><td>0x0001</td><td>Set States</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
<tr><td>State</td><td>0xF30A</td><td>Upgrade Option</td></tr>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
<tr><td>Value</td><td></td><td>Upgrade Option
0x00 - Disable OTA Upgrades
0x01 – Enable OTA Upgrades</td></tr>
</table></td></tr>
<tr><td>Notes</td><td></td></tr>
</table>

Avatar Upgrade Status

<table>
<tr><td>Message</td><td>Upgrade Status</td></tr>
<tr><td>Direction</td><td>Avatar -> Device</td></tr>
<tr><td></td><td>This is used to inform the Device on the state of the Avatar upgrade process.

<table>
<tr><td colspan="3">Request : Device->Avatar</td></tr>
<tr><td>Message</td><td>0x0001</td><td>Set States</td></tr>
<tr><td>Count</td><td>1</td><td></td></tr>
</table></td></tr>
</table>

	<table><tr><td>State</td><td>0x030B</td><td>Avatar Upgrade Status</td></tr><tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr><tr><td>Value</td><td></td><td>Upgrade Statuses See table below</td></tr></table> Avatar Upgrade Statuses <table><tr><td>0x00</td><td>Avatar Upgrade available.</td><td>An Avatar only upgrade downloaded.</td></tr><tr><td>0x01</td><td>Avatar and Device upgrade available.</td><td>An upgrade containing software for both the Avatar and Device downloaded.</td></tr><tr><td>0x02</td><td>Device upgrade available.</td><td>An upgrade containing software for the Device only downloaded.</td></tr><tr><td>0x03</td><td>Start Device Upgrade</td><td>Indicates to the device that it should begin the data transfer / upgrade immediately.</td></tr><tr><td>0x04</td><td>Avatar Upgrade in progress.</td><td>Indicates that the Avatar is performing an upgrade of its own firmware.</td></tr><tr><td>0x05</td><td>Avatar Upgrade complete.</td><td>Indicates that the Avatar has completed its firmware upgrade</td></tr><tr><td>0x06</td><td>Avatar Upgrade fail.</td><td>Indicates that there we problems with the Avatar firmware upgrade.</td></tr></table>
Notes	Upgrade statuses 0x00 - 0x02 will be used where the Upgrade Option is Silent, or Use controlled to indicate that an Upgrade is available/scheduled. Upgrade status 0x03 will be used with Forced and Silent Upgrade Options to indicate that the upgrade should begin immediately. Upgrade statuses 0x04 – 0x07 will be used during the Avatar upgrade firstly to mute outputs on the device, and secondly to give a visual indication (where appropriate) of the upgrade progress.

Upgrade Data

Message	Upgrade Data
Direction	Avatar -> Device
	Similar to the functions mentioned in the Citation document (section 4)

<table>
<tr><td></td><td>This is used to send upgrade data to the device.

Due to the size of the upgrade, images this will be repeated a number of times to send the entire upgrade data. All messages for a single transfer will have the same frame id and incrementing packet ids.

<table>
<tr><th colspan="3">Request : Device->Avatar</th></tr>
<tr><td>Message</td><td>0x0001</td><td>Set States</td></tr>
<tr><td>Count</td><td>3</td><td></td></tr>
<tr><td>State</td><td>0x030C</td><td>Upgrade Data Size</td></tr>
<tr><td>Encoding</td><td>0xCE</td><td>UINT32</td></tr>
<tr><td>Value</td><td></td><td>The total size of the upgrade data.</td></tr>
<tr><td>State</td><td>0x030D</td><td>Upgrade Data Offset</td></tr>
<tr><td>Encoding</td><td>0xCE</td><td>UINT32</td></tr>
<tr><td>Value</td><td></td><td>The offset in the upgrade data of the data contained in the packet.</td></tr>
<tr><td>State</td><td>0x030E</td><td>Upgrade Data</td></tr>
<tr><td>Encoding</td><td>0xC5</td><td>BIN16</td></tr>
<tr><td>Value</td><td></td><td>The upgrade data.</td></tr>
</table>
</td></tr>
<tr><td>Notes</td><td>The Citation message includes an offset which may be useful, but is not completely necessary if the packets are numbered sequentially, but has been included here

Stream use a default data size of 4096 bytes.</td></tr>
</table>

Device Upgrade Status

<table>
<tr><td>Message</td><td>Upgrade Status</td></tr>
<tr><td>Direction</td><td>Device -> Avatar</td></tr>
<tr><td></td><td>Similar to the functions mentioned in the Citation document (section 4.5)

This will be used to inform the Avatar that the device has finished upgrading, or to inform it that the check sum failed and the upgrade has been abandoned.

<table>
<tr><th colspan="3">Request : Device->Avatar</th></tr>
<tr><td>Message</td><td>0x0001</td><td>Set States</td></tr>
<tr><td>Count</td><td>1 / 2</td><td>Dependent on optional field</td></tr>
<tr><td>State</td><td>0xF30B</td><td>Device Upgrade Status</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>Encoding</td><td>0xCC</td><td>UINT8</td></tr>
<tr><td>Value</td><td></td><td>Upgrade Status
0x00 Ready to Receive Data
0x01 Data Received
0x02 Transfer Successful
0x03 Checksum error
0x04 Upgrade starting
0x05 Upgrade Complete
0x06 Upgrade Fail
0x07 n/a
0x08 Start Avatar USB Upgrade</td></tr>
<tr><td>State</td><td>0xF30D</td><td>Device Upgrade Jump To Offset (Optional)</td></tr>
<tr><td>Encoding</td><td>0xCE</td><td>UINT32</td></tr>
<tr><td>Value</td><td></td><td>An offset from the start of the Upgrade Data (including any checksums/metadata)

This is an optional field</td></tr>
</table>
</td></tr>
<tr><td>Notes</td><td>Status 0x00 indicates that the Device is ready to receive upgrade data from the Avatar

Note: The device will have triggered a reboot between receiving the Upgrade data and signalling that the Upgrade is complete.

<table>
<tr><td>0x08</td><td>Start Avatar USB Upgrade</td><td>Reboots into USB upgrade mode and attempts an upgrade from image.swu if present on the USB stick</td></tr>
</table>
</td></tr>
</table>

Questions / Notes
• Note: We assume that the handshaking process will be completed before the Upgrade process begins.