4.5 Characteristics for Hearing Aid Control Service

4.5.1 Hearing aid configuration characteristic

App will need to read this on connect (startup or reconnect).

Hearing aid configuration (R) – UUID dcafb8**01-…**			
Byte length	**Short description**	**Type**	**Comments**
1	Speech language	Uint8	Language configured during fitting session.
2	Fitting number	Uint16	If number is different since app read it last time, then the fitting has changed. Program configuration might have changed. If the version reaches 65.535 the next value will be 0. App needs to check if it is different (not greater than).
1	Program count	Uint8	The number of programs. This value is needed for selecting and reading the Program configuration.
1	Stream type count	Uint8	The number of supported stream types. This value is needed for selecting and reading the Stream type configuration.
1	Streaming volume Index count	Uint8	The number of Streaming volume Indexes used for volume/mute
1	Microphone volume steps	Uint8	Total number of steps for Microphone volume.
1	Streaming volume steps	Uint8	Total number of steps for Streaming volume.
1	Default microphone volume	Uint8	Lowest value is 0. Default volume tells the default level.
1	Default streaming volume	Uint8	Lowest value is 0. Default volume tells the default level.
16	User-ID	16 * Uint8	GDPR client id from fitting session.
1	Headset mode allowed	Bool	True if FitXP is allowing headset mode to be possible.
1	Microphone equalizer Index count	Uint8	The number of Microphone equalizer Indexes used.
1	Microphone volume index count	Uint8	The number of microphone volume indexes used.

1	Personal programs count	Uint8	The number of personal programs that can be stored in the HA.
1	Index of microphone volume for first personal program	Uint8	Index into Microphone volume characteristic to use for first personal program. Each personal program will have a dedicated volume.
1	Index of microphone equalizer for first personal program	Uint8	Index into Microphone Equalizers Characteristic to use for first personal program. Each personal program will have a dedicated set of equalizer settings.
1	Streaming equalizer index count	Uint8	The collected number of streaming equalizer indexes used for both music and speech streaming
1	Music Streaming Mode available	Bool	Boolean that represents the availability of Music Streaming Mode (MSM). True = MSM available
1	Is demo HA	Bool	Boolean telling if the HA is a Demo HA or not
1	Demo Type Variant	Uint8	Type variant for Demo HA Copied from Definfo: Used to indicate current Type Variant for Demo-HAs
1	HA Type Variant	Uint8	Type variant of the HA Copied from Definfo: Bestsellers Variant. Extern-Channel/bands. (0=Not_defined.1=440,2=330,3=220,4=110, 5=100, 6=50, 7=30, 8+ not used
1	HA Type Variant Converted	Uint8	Converted type variant of HA. Copied from Definfo: HA-Serie variant last converted by fitting. (0=Not_defined. 1=440, 2=330, 3=220, 4=110 5-7 not used
1	AI enabled	Bool	Boolean telling if the HA is an AI model or not

4.5.2 Select program configuration characteristic

Provide the index of the program that will be read using Program configuration characteristic.

Select program configuration (R/W) – UUID dcafb8**02-…**			
Byte length	**Short description**	**Type**	**Comments**
1	Index	Uint8	Write an index between 0 and program count minus 1. Then read Program configuration characteristic to retrieve information about that program.

After writing which program to read, then read Program configuration characteristic to get data about the selected program.

Returns the ATT error code ValueNotAllowed = 0x13 on write if index is invalid.

4.5.3 Program configuration characteristic

This characteristic is used to read information about a program. It will provide the name and other program settings depending on the program type. This provides information about programs configured at fitting.

Returns the information about the program that was requested in the write command.

Program configuration (R) – UUID dcafb8**03-…**			
Byte length	**Short description**	**Type**	**Comments**
1	Program index	Uint8	Position of the program in SYSCTRL_FW_UI_ProgramList. Value is between 0 and program count minus 1.
1	Program template	Uint8	Value from FIT_DB_Prog_Template for the program
1	Program icon	Uint8	Index to icon to display for the program.
1	Microphone equalizer index	Uint8	Tells which microphone equalizer index to use. For the Telecoil program, the value is 255 if SYSCTRL_FW_TelecoilMicDisable is set to 1.
28	Program name	String	Program name UTF-8. 0-terminated. No 0-termination if program name uses all 28 bytes.
1	Program key	Uint8	Value of SYSCTRL_FW_UI_ProgramList at “Program index”

4.5.4 Stream type sound setting indexes characteristic

The sound setting indexes incudes volume indexes and equalizer indexes used for reading or updating volumes, mutes and equalizers (for both speech streaming and music streaming when relevant). Two or more stream types can share the same volume index or equalizer indexes.

Stream type sound setting indexes (R) – UUID dcafb8**04-…**			
Byte length	**Short description**	**Type**	**Comments**
Stream type count x 3	Streaming indexes	Uint8 x 3	3 values for each stream type: 1. byte = volume index for volume and mute 2. byte = equalizer index for speech streaming mode 3. byte = equalizer index for music streaming mode. If MSM is not valid for stream type or disabled for HA, then value = 255

Example below limited to 3 stream types instead of all to keep it simple.

Byte	Data - Example	Description
0	0	Volume index for volume and mute = 0 for Stream type 0
1	0	Equalizer index for speech streaming = 0 for Stream type 0
2	1	Equalizer index for music streaming = 1 for Stream type 0
3	1	Volume index for volume and mute = 1 for Stream type 1
4	2	Equalizer index for speech streaming = 2 for Stream type 1
5	3	Equalizer index for music streaming = 3 for Stream type 1
6	2	Volume index for volume and mute = 2 for Stream type 2
7	4	Equalizer index for speech streaming = 4 for Stream type 2
8	255	Equalizer index for music streaming = Not Valid for Stream type 2

4.5.5 Personal program characteristic

Personal program (R/W) – UUID dcafb8**06-…**			
Byte length	**Short description**	**Type**	**Comments**
1	Program key	Uint8	Key of the currently accessed personal program between 0x50 to 0x54 both included. Reflects the key value of the written index to “Select personal program characteristic”. When writing, this field selects the personal program to overwrite.
1	Parent program	Uint8	Key of the program on which the personal program is based.
1	Program template	Uint8	Program template of the parent program. This is needed to remove invalid personal programs after refitting.
1	Program icon	Uint8	Index to icon to display for the program
28	Name	Uint8 array	UTF-8 encoded name. Unused positions should be set to zero, but zero-termination is not required.
1	Volume	Uint8	Stored volume of the personal program. The format follows the format of Microphone volume characteristic and includes mute.
3	Equalizer settings	3 x Int8	Stored and applied Microphone equalizer settings. The format follows the format of a single entry in Microphone equalizers characteristic.
1	Fast-compressor setting	Uint8	Stored and applied fast compressor setting. Value 255: indicates the fast compressor setting is not applied
1	Selectable from local controls	Uint8	When set, the program will be included in the selection sequence that the user can access via the program button and the RC-Dex.

Writing to volume or equalizer-setting of any personal program will update the corresponding values in “Microphone volume characteristic” and/or “Microphone equalizer characteristic”. This will result in notifications if the values change.

4.5.6 TV Play configuration characteristic

TV Play (R/W/N) – UUID dcafb8**07-…**			
Byte length	**Short description**	**Type**	**Comments**
1	TV Play 1 valid	Uint8	Boolean which represents if the associated TV play is valid or not (a value different from 0 means valid)
1	TV Play 1, Program Key	Uint8	Program key for the first TV Play
1	TV Play 1, HW revision	Uint8	HW revision of the first TV Play
4	TV Play 1, FW version	Uint8 x 4	FW version of the first TV Play
3	TV Play 1, Serial Number	Uint8 x 3	Serial number of the first TV Play
1	TV Play 1, Index of microphone volume	Uint8	Index into Microphone Volume Characteristic for first TV Play.
1	TV Play 1, Index of microphone equalizer	Uint8	Index into Microphone equalizer settings Characteristic for the first TV Play.
1	TV Play 2 valid	Uint8	Boolean which represents if the associated TV play is valid or not (a value different from 0 means valid)
1	TV Play 2, Program Key	Uint8	Program key for the second TV Play
1	TV Play 2, HW revision	Uint8	HW revision of the second TV Play
4	TV Play 2, FW version	Uint8 x 4	FW version of the second TV Play
3	TV Play 2, Serial Number	Uint8 x 3	Serial number of the second TV Play
1	TV play 2, Index of microphone volume	Uint8	Index into Microphone Volume Characteristic for second TV play.
1	TV play 2, Index of microphone equalizer	Uint8	Index into Microphone equalizer Characteristic for second TV play.
1	TV Play 3 valid	Uint8	Boolean which represents if the associated TV play is valid or not (a value different from 0 means valid)
1	Tv Play 3, Program Key	Uint8	Program key for third TV play

TV Play (R/W/N) – UUID dcafb8**07-…**			
1	TV Play 3, HW revision	Uint8	HW revision of the third TV Play
4	TV Play 3, FW version	Uint8 x 4	FW version of the third TV Play
3	TV Play 3, Serial Number	Uint8 x 3	Serial number of the third TV Play
1	TV play 3, Index of microphone volume	Uint8	Index into Microphone Volume Characteristic for third TV play.
1	TV play 3, Index of microphone equalizer	Uint8	Index into Microphone equalizer Characteristic for third TV play.

For release 3 only one TV Play connection is supported so 0's will always be returned for number 2 and 3. App needs to be able not to fail if eSW starts supporting it. App does not need to build support at the same time but simply be able to ignore support for more than one until the App is ready.

Writing to this characteristic is only used for deleting TV Play pairings. i.e. writing a '0' to the "TV Play X valid" field will remove the corresponding TV Play pairing.

When writing to this characteristic all fields must be filled out with correct values.

Example read:

Byte	**Data - Example**	**Description**
0	0x01	TV Play 1 valid
1	0x3C	TV Play 1 Program key
2	0x01	TV Play 1 HW Revision: "Rev. 1"
3..6	0x01 0x00 0x10 0x00	TV Play 1 FW Version: "FW version 1.0.16"
7..9	0x12 0x11 0x10	TV Play 1 Serial Number "SN: 1052946"
10	0x00	TV play 1 Microphone volume index: 0
11	0x00	TV play 1 Microphone equalizer index: 0
12	0x01	TV Play 2 valid
13	0x3D	TV play 2 Program key
14	0x01	TV Play 2 HW Revision: "Rev. 1"
15..18	0x01 0x00 0x01 0x00	TV Play 2 FW Version: "FW version 1.0.16"
19..21	0x13 0x11 0x10	TV Play 2 Serial Number "SN: 1052947"
22	0x00	TV Play 2 Microphone volume index: 0

23	0x01	TV Play 2 Microphone equalizer index: 1
24	0x01	TV Play 3 valid
25	0x3E	TV Play 3 Program key
26	0x01	TV Play 3 HW Revision: “Rev. 1”
27..30	0x01 0x00 0x10 0x00	TV Play 3 FW Version: “FW version 1.0.16”
31..33	0x14 0x11 0x10	TV Play 3 Serial Number “SN: 1052948”
34	0x00	TV Play 3 Microphone volume index: 0
35	0x02	TV Play 3 Microphone equalizer index: 2

4.5.7 TV Play custom name characteristic

TV Play custom name (R/W) – UUID dcafb8**08-…**			
Byte length	**Short description**	**Type**	**Comments**
28	TV Play 1 Program name	Uint8	Read/write custom TV play name
28	TV Play 2 Program name	Uint8	Read/write custom TV play name
28	TV Play 3 Program name	Uint8	Read/write custom TV play name

Characteristic to enable the App to write custom names to the TV play programs. The strings are ASCII encoded and must be '0' terminated.

Example:

Byte	**Data - Example**	**Description**
0..27	"My TV Play 1"\0	TV play 1 Program name
28..55	"My TV Play 2"\0	TV play 2 Program name
56..83	"My TV Play 3"\0	TV play 3 Program name

4.5.8 Battery status characteristic

Returns battery information.

Battery status (R/N) – UUID dcafb8**0a-…**			
Byte length	**Short description**	**Type**	**Comments**
1	Battery remaining	Uint8	Percentage of battery remaining. The percentage is stored in the 7 least significant bits. The most significant bit indicates if valid (0) or invalid (1). The measure will be invalid for a period after boot.
2	Cycles	Uint16	How many times the battery has been cycled. Only relevant for rechargeable HAs. 0 is always returned if not a rechargeable battery.

4.5.9 Microphone volumes characteristic

Microphone volumes and mutes.

Microphone volumes (R/W/N) – UUID dcafb8**0b-…**			
Byte length	**Short description**	**Type**	**Comments**
Number of microphone volumes indexes	Microphone volume and mute	Uint8	Volume between 0 and Microphone steps from Hearing aid characteristic minus 1. Lowest number is lowest volume. Volume is stored in the 7 least significant bits. The most significant bit indicates if muted (1) or not (0).

Returns the ATT error code ValueNotAllowed = 0x13 on write if volume is invalid.

Example 1

Byte	**Data - Example**	**Description**
0	0x00 = 0b**0**0000000	Microphone volume 0 and unmuted for Microphone

Example 2

Byte	**Data - Example**	**Description**
0	0x84 = 0b**1**0000100	Microphone volume 4 and muted for Microphone

4.5.10 Streaming volumes characteristic

Table of all streaming volumes and mutes. Each streaming volume index has one byte of information. To update one or more volumes, the corresponding byte needs to be modified.

Streaming volumes (R/W/N) – UUID dcafb8**0c-…**			
Byte length	**Short description**	**Type**	**Comments**
1~~Number of streaming volume indexes~~ ~~(read in Hearing Aid Configuration Characteristic)~~	Streaming volume 0~~Streaming volume and mute~~	Uint8	Volume for streaming volume index 0 between 0 and 255. Lowest number is lowest volume. ~~Volume between 0 and Streaming steps from Hearing Aid Configuration Characteristic minus 1. Lowest number is lowest volume.~~ ~~Volume is stored in the least significant 7 bits. The most significant bit indicates if muted (1) or not (0).~~ Volumes are super sticky (i.e., persisted over reboot). MFi and ASHA sets the volume when a stream is starting.
1	Streaming mute 0	Bool	True if streaming volume index 0 is muted.
1	Streaming volume 1	Uint8	Volume for streaming volume index 1
1	Streaming mute 1	Bool	True if streaming volume index 1 is muted.
…			
1	Streaming volume n	Uint8	Volume for streaming volume index n
1	Streaming mute n	Bool	True if streaming volume index n is muted.

Formatted Table

n = Number of streaming volume indexes (read in Hearing Aid Con-figuration Characteristic).

Example (Assuming only four Volume Indexes for Streaming):

Byte	Data - Example	Description

0	0~~x00 = 0b**0**0000000~~	Streaming volume 0~~ and unmuted~~ for Streaming Volume Index = 0.
1	0	Streaming is not muted for Streaming Volume Index = 0
2~~1~~	~~0x84 = 0b**1**00001004~~	Streaming volume 4~~ and muted~~ for Streaming Volume Index = 1.
3	1	Streaming is muted for Streaming Volume Index = 1
4~~2~~	~~0x0A = 0b**0**000101010~~	Streaming volume 10~~ and unmuted~~ for Streaming Volume Index = 2.
5	0	Streaming is not muted for Streaming Volume Index = 2
6~~3~~	~~0x80 = 0b**1**00000000~~	Streaming volume 0~~ and muted~~ for Streaming Volume Index = 3.
7	1	Streaming is muted for Streaming Volume Index = 3

Returns the ATT error code InvalidAttributeLen = 0x0D if wrong number of volumes and mutes are provided.

~~Returns the ATT error code ValueNotAllowed = 0x13 on write, if one or more volumes are invalid.~~

How to convert from full volume to steps and vice versa

Conversion from full volume (between 0 and 255) into volume steps can be done by calling ConvertRanges(volume_in_full_range, 255, number_of_streaming_volume_steps - 1)
Conversion from steps to full volume can be done by calling ConvertRanges(volume_in_steps, number_of_streaming_volume_steps - 1, 255)
The number of steps are read from Hearing Aid Configuration characteristic.

Here is the function we use to calculate in the firmware. It is important that app does the same conversion. Otherwise there will be a mismatch when using HA buttons or RC Dex to change volume.

```
/**
 * @brief Convert value from one scale to another
 *
 * Method uses 16,16 bit fractional numbers for high precision when converting from
 * large range scale to small range scale.
 * Method assumes valueIn is zero based, if converting from/to scales not having
 * minimum at 0, the input/output values must be normalized to/from minimum at 0
```

```
 * before/after this method.
 *
 * @param[in] valueIn          Volume value to be converted to a different scale, value is
 *                             expected to be zero based
 * @param[in] volRangeIn       Range of scale for input value (max value - min value)
 * @param[in] volRangeOut      Range of scale for output value (max value - min value)
 * @return                     Re-scaled volume corresponding to input volume. 0 if
 *                             volRangeIn = 0
 */
uint16_t ScaleConverter::ConvertRanges(uint8_t valueIn, uint8_t volRangeIn, uint8_t
volRangeOut) {
  if (0 == volRangeIn) {  // Detect and avoid division by zero
    return 0;
  }

  uint32_t fracVolRangeOut = volRangeOut << 16;
  uint32_t fracConversionFactor = fracVolRangeOut / volRangeIn;

  uint32_t fracVolOut = valueIn * fracConversionFactor;
  fracVolOut += 1 << 15;  // Rounding increment, if incrementing to round is incorrect,
                          // added bit will be shifted out when removing fractional part
  uint16_t volOut = fracVolOut >> 16;
  return volOut;
}
```

~~4.5.12~~4.5.11 Microphone equalizers characteristic

Table of all microphone equalizer settings (bass, middle and treble). The hearing aid will update the microphone equalizers.

Microphone equalizers (R/W/N) – UUID dcafb8**0d-…**			
Byte length	**Short description**	**Type**	**Comments**
1	Bass 0	Int8	Allowed values range from -6 to +6 0 is the neutral value.
1	Middle 0	Int8	Allowed values range from -6 to +6 0 is the neutral value.
1	Treble 0	Int8	Allowed values range from -6 to +6 0 is the neutral value.
1	Bass 1	Int8	Allowed values range from -6 to +6 0 is the neutral value.
1	Middle 1	Int8	Allowed values range from -6 to +6 0 is the neutral value.
1	Treble 1	Int8	Allowed values range from -6 to +6 0 is the neutral value.
…	…	…	… This continues until the number of Microphone equalizer indexes has been met (called n below).
1	Bass n	Int8	Allowed values range from -6 to +6 0 is the neutral value.
1	Middle n	Int8	Allowed values range from -6 to +6 0 is the neutral value.
1	Treble n	Int8	Allowed values range from -6 to +6 0 is the neutral value.

Example (Assuming only four equalizer Indexes for Microphone):

Byte	**Data - Example**	**Description**
0	0x01	Bass 1 for Microphone equalizer Index = 0.
1	0x02	Middle 2 for Microphone equalizer Index = 0.

2	0x03	Treble 3 for Microphone equalizer Index = 0.
3	0XFA	Bass -6 for Microphone equalizer Index = 1.
4	0x00	Middle 0 for Microphone equalizer Index = 1.
5	0x06	Treble 6 for Microphone equalizer Index = 1.
6	0x04	Bass 4 for Microphone equalizer Index = 2.
7	0x05	Middle 5 for Microphone equalizer Index = 2.
8	0x06	Treble 6 for Microphone equalizer Index = 2.
9	0x06	Bass 6 for Microphone equalizer Index = 3.
10	0x05	Middle 5 for Microphone equalizer Index = 3.
11	0x04	Treble 4 for Microphone equalizer Index = 3.

Returns the ATT error code InvalidAttributeLen = 0x0D if wrong number of equalizers are provided.

Returns the ATT error code ValueNotAllowed = 0x13 on write, if one or more equalizer values are invalid.

~~4.5.13~~4.5.12 Streaming equalizers characteristic

Table of all streaming equalizer settings (bass, middle and treble) for speech streaming and music streaming. The hearing aid will update the streaming equalizers.

Streaming equalizers (R/W/N) – UUID dcafb8**0e-…**			
Byte length	**Short description**	**Type**	**Comments**
1	Bass 0	Int8	Allowed values range from -6 to +6 0 is the neutral value.
1	Middle 0	Int8	Allowed values range from -6 to +6 0 is the neutral value.
1	Treble 0	Int8	Allowed values range from -6 to +6 0 is the neutral value.
1	Bass 1	Int8	Allowed values range from -6 to +6 0 is the neutral value.
1	Middle 1	Int8	Allowed values range from -6 to +6 0 is the neutral value.
1	Treble 1	Int8	Allowed values range from -6 to +6 0 is the neutral value.
…	…	…	… This continues until the number of Streaming equalizer indexes has been met (called n below).
1	Bass n	Int8	Allowed values range from -6 to +6 0 is the neutral value.
1	Middle n	Int8	Allowed values range from -6 to +6 0 is the neutral value.
1	Treble n	Int8	Allowed values range from -6 to +6 0 is the neutral value.

See example for microphone equalizers in the previous section.

Returns the ATT error code InvalidAttributeLen = 0x0D if wrong number of equalizers are provided.

Returns the ATT error code ValueNotAllowed = 0x13 on write, if one or more equalizer values are invalid.

~~4.5.14~~4.5.13 Active program characteristic

On write, the hearing aid will switch to the program requested. Read/Notify will inform about current settings.

Active program (R/W/N) – UUID dcafb8**0f-…**			
Byte length	**Short description**	**Type**	**Comments**
1	Program key	Uint8	Current active program

Returns the ATT error code ValueNotAllowed = 0x13 on write if the program is illegal.

The program key has the following interpretation:

0 - 20	HCP programs described by the Program configuration characteristic. The value is the *key* element from said characteristic.
40 - 47	Directional focus 40 / 41: Focus front / focus front with boost 42 / 43: Focus rear / focus rear with boost 44 / 45: Focus left / focus left with boost 46 / 47: Focus right / focus right with boost
60 - 62	TV Play 0, 1 and 2
80 - 84	Personal programs
100 – 131	Special programs used for internal testing. It is not expected that the App shall support these in any way.

~~4.5.15~~4.5.14 Stream status characteristic

Any stream that starts or stops will trigger a notification. App can change Streaming Mode.

Stream status (R/W/N) – UUID dcafb8**10-…**			
Byte length	**Short description**	**Type**	**Comments**
1	Playing stream type	Uint8	Currently playing stream type. When writing a new streaming mode this must be the value for current stream.
4	Active stream types	Uint32	Bit field of all currently active stream types. Only the one mentioned in “Playing stream type” is being played. See Stream Types in section 4.4
1	Streaming Mode	Uint8	Current streaming Mode. Speech = 0, Music = 1, Not relevant = 2. App can be either notified or request another mode (Speech or Music)

Returns the ATT error code ValueNotAllowed = 0x13 on write if the playing stream type is no longer active and the Streaming Mode is therefore not applied.

Example:

Byte	**Data - Example**	**Description**
0	0x01	Stream type 1 is currently being played
1..4	0x02 0x03 0x04 0x00	0x00040302 = 0b0000 0000 0000 0100 0000 0011 0000 0010 Each of the 32 bits tells if a stream is available. In the example above, the bits for Stream type 1, 8, 9 and 18 are set, so these streams are available.
5	0x01	App is notified that Streaming Mode = Music. App selects Streaming Mode = Music

~~4.5.16~~4.5.15 Bluetooth off characteristic

Request to turn off the Bluetooth chip.

Bluetooth off (R/W) – UUID dcafb8**11-…**			
Byte length	**Short description**	**Type**	**Comments**
1	Turn off Bluetooth	Bool	Set true to turn it off.
2	Timeout	Uint16	The number of minutes Bluetooth should be off before turning it automatically on again. 0 means off forever (until reboot).

~~4.5.17~~4.5.16 Play indication characteristic

A request to play an indication (e.g., a sound).

Play indication (R/W) – UUID dcafb8**12-…**			
Byte length	**Short description**	**Type**	**Comments**
1	Indication	Uint8	No indication = 0 Lowest volume reached = 1 Middle volume reached = 2 Highest volume reached = 3 Volume increased = 4 Volume decreased = 5 Mute toggled = 6 Bluetooth off = 7 New indications can be added if needed. E.g., Sounds or LED indications.

Returns the ATT error code ValueNotAllowed = 0x13 on write if the Indication is illegal.

~~4.5.18~~4.5.17 Headset mode configuration characteristic

Characteristic to enable or disable headset mode from the app.

Headset mode configuration (R/W) – UUID dcafb8**13-…**			
Byte length	**Short description**	**Type**	**Comments**
1	Headset mode enabled	Bool	If true and Headset mode allowed from Hearing Aid Configuration Characteristic is also true, then the Hearing Aid will inform iPhones that Headset Mode is supported. Also used for LE Audio in the future.
1	Double tap enabled	Bool	If true, then double tap can be used to accept and incoming call and end an active call. If false, then only button and phone can control the calls. Not the motion sensor.

To enable headset function, it should be enabled both in FitXP and in the APP.
If headset function is enabled the HAW can always control the call on the phone or via the push button on the HA.
In addition, if the double tab feature is enabled, the call can also be controlled by double tab.
If headset function is disabled, the call can only be controlled on the telephone.
Headset function is only available on phones supporting LEA2.
If headset function is disabled, even a phone supporting LEA2 will work like a pre-LEA2 phone.

~~4.5.19~~4.5.18 Select personal program characteristic

Select personal program configuration (W) – UUID dcafb8**14--…**			
Byte length	**Short description**	**Type**	**Comments**
1	Index	Uint8	Write an index between 0 and personal program count minus 1. Then read personal program characteristic to retrieve information about that program.

~~4.5.20~~4.5.19 Personal program ordering characteristic

Table of all completed personal programs and their order. If a personal program is mentioned in this list, it is assumed to be valid and complete.

Personal program ordering (R/W) – UUID dcafb8**15--…**			
Byte length	**Short description**	**Type**	**Comments**
4	Sequence number	Uint32	This sequence number is fully controlled by the App and is meant to be used for detecting and resolving inconsistencies between left and right HA.
N	Array of personal-program keys.	Uint8 x N	Array of keys, starting from 0x50 to 0x54 both included. The number of entries (N) will match the number of personal program entries reported by the HAC.

Example (3 personal programs and two empty slots)

Byte	**Data example**	**Description**
0..3	317	Sequence number
4	0x51	Personal program 1
5	0x50	Personal program 0
6	0x52	Personal program 2
7	0xFF	Empty
8	0xFF	Empty

To delete a personal program, the App must simply write an array of keys that does not include the program to be deleted.

A specific personal program should not be added to the list before relevant data has been written to the Personal Program Characteristic.

Due to the sequence number in the beginning, the App must always write the entire characteristic.

If invalid values are included in the write, an error will be returned.

In a brand-new HA, where no Personal Program settings are configured, the firmware will initialize empty ordering list (with key 0xFF) and set sequence number to “0” during the first boot.

If, at boot, the HA determines that the information in a corresponding Personal Program is inconsistent, then the personal program will be removed from the list and sequence number will be set to “0”. The heuristics for determining this are as follows:

- The template of the parent-program and the registered program template do not match.
- The parent-program is no longer configured in the HA.

This is not bulletproof, and the HCP can easily make changes to the configuration that will render the personal program obsolete, without being caught by the heuristic.

The sequence number should always be increased each time the App writes to the ordering characteristic. If the App only succeeds in writing to one of the HA’s, then this value will differ, and the App can easily update the other HA. This of course requires the App to write to the Personal Program Characteristic first in both HA’s before writing the ordering to one HA at a time.

The HA may or may not resolve the situation itself, so the App should be ready to resolve any inconsistencies.

~~4.5.22~~4.5.21 Reset Sound characteristic

Provide the index of the program to reset microphone volume, mute and equalizer.

Also provide stream type. If stream type is none then ignore it. Otherwise reset the streaming volume, mute and equalizer for that stream type.

Reset Sound (R/W) – UUID dcafb8**16-…**			
Byte length	**Short description**	**Type**	**Comments**
1	ProgramKey	Uint8	Microphone volume, mute and equalizer are reset for the program key provided. Volume and mute are global for all programs. Equalizer for the microphone equalizer index related to the program key will be reset. If more programs share the same microphone equalizer index, then all of them are reset.
1	StreamType	Uint8	Provide the stream type. Type 0 is none. If something else than none is provided, then the volume, mute and equalizer will be reset for that stream type. If more stream types share the same streaming sound setting index, then it affects all of them.

Returns the ATT error code ValueNotAllowed = 0x13 on write if program key or stream type is not allowed.

This characteristic can affect both Microphone Volumes characteristic and Microphone Equalizers characteristic. They will notify if the reset results in changes.

The same applies for Streaming Volumes characteristic and Streaming Equalizers characteristic if stream type is not none.

~~4.5.23~~4.5.22 Sound Assist characteristic

Sound Assist (R/N) – UUID dcafb817-…			
Byte length	**Short description**	**Type**	**Comments**
1	Firmware ID, Major	Uint8	Firmware ID of Sound Assist – Major version. 0 if not paired
1	Firmware ID, Minor	Uint8	Firmware ID of Sound Assist – Minor version
2	Firmware ID, Patch	Uint16	Firmware ID of Sound Assist – Patch number
1	Hardware ID	Uint8	Hardware ID of Sound Assist. 0 if not paired

Example:

Byte	Data - Example	Description
0	0x01	Firmware ID: Major version = 0x01
1	0x02	Firmware ID: Minor version = 0x02
2..3	0x03 0x04	Firmware ID: Patch 0x0403
4	0x2A	Hardware ID: 0x2A

4.6 Maintenance Service

4.6.1 Persistent logging characteristic

This characteristic reads all persisted log messages in one blob.

Persistent logging (R) – UUID b6f3df**01**-...			
Byte length	**Short description**	**Type**	**Comments**
4	FirstLogId	Uint32	First log id for the pool of log entries in the message, see [4]
Variable	Log messages	Blob	One blob of data is returned. App does not need to interpret the data, but simply provide them as is to the cloud (in app log or otherwise)

App needs to inform the cloud about the messages and that they were from persistent log.

For information but not something app shall handle:

Log messages will contain information of whether the message is from BLE, SysCTRL or TCTRL.

Log messages on level ERROR, FATAL and FORCED can be available.

Log messages are in a packed format.

4.6.1.1 Read persistent using APP

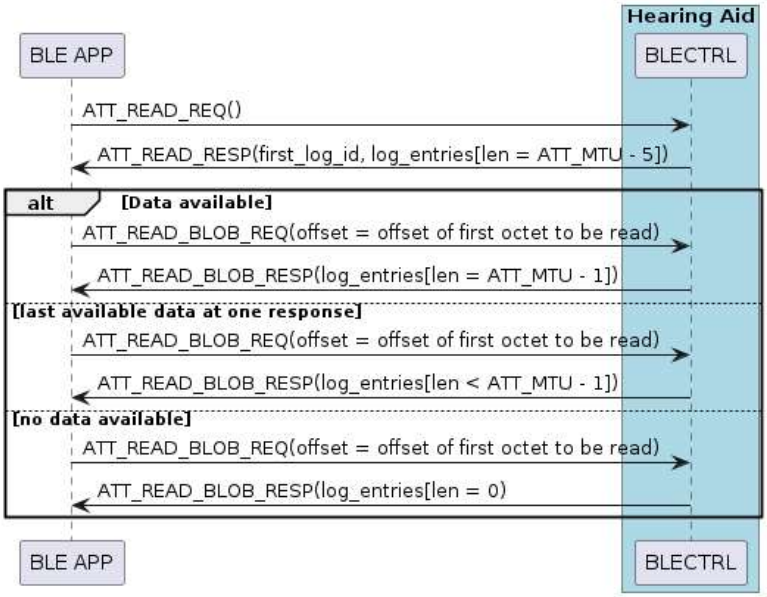
APP read logs over BLE.

Below are the steps for the APP to read logs from HA persistent log:

1. APP soft shall request GATT read request to HA with persistent logging handleID.
2. HA shall process this read request and fetch logs from persistent memory.
3. HA shall form a single blob packet of the logs.
4. HA sends logs to APP over BLE read ATT response

Blob Maximum Length - Maximum blob packet size is ATT_MTU - 1

Diagram: Reading persistent logs



4.6.2 Event logging characteristic

This characteristic provides one or more logs in one blob message inform of notification.

Event logging (R/N) – UUID b6f3df**02**-...			
Byte length	**Short description**	**Type**	**Comments**
Variable	Log messages	Blob	One blob of data is returned. App does not need to interpret the data, but simply provide them as is to the cloud (in app log or otherwise)

App needs to inform the cloud about the message, timestamp of receiving, and that it is from the event log. It would be nice to add them inside the app log to provide information about under which circumstances they occurred.

For information but not something app shall handle:

The log message will contain information of whether the message is from BLE, SysCTRL or TCTRL.

Log messages on level INFO, WARNING, ERROR, FORCED can be available.

Log messages are in a packed format.

4.6.3 Change log level characteristic

This characteristic can set the log level reported in Event logging characteristic.
This is used for debugging purposes.

Change log level (R/W) – UUID b6f3df**03**-...			
Byte length	**Short description**	**Type**	**Comments**
1	Log level	Uint8	None = 0, Forced = 1, Fatal = 2, Error = 3, Warning = 4, Information = 5, Debug = 6. Normally log level should be at least 3. Default is 5.

Returns the ATT error code ValueNotAllowed = 0x13 on write if the Log level is illegal.

4.6.4 Firmware version characteristic

Currently running firmware version.

Firmware version (R) – UUID b6f3df**06**-...			
Byte length	**Short description**	**Type**	**Comments**
1	Firmware version, Major	Uint8	Major version of the currently running firmware.
1	Firmware version, Minor	Uint8	Minor version of the currently running firmware.
2	Firmware version, Build	Uint16	Build version of the currently running firmware.
1	App interface version, Major	Uint8	Major version of the App interface.
1	App interface version, minor	Uint8	Minor version of the App interface.
1	Fitting interface version, major	Uint8	Major version of the fitting and service interface.
1	Fitting interface version, minor	Uint8	Minor version of the fitting and service interface.
1	Forced minimum version	Uint8	Version identifier to force compatible upgrade.
1	AI image version, major	UInt8	Major version of the currently running AI image.
1	AI image version, minor	UInt8	Minor version of the currently running AI image.
2	AI image version, build	UInt16	Build version of the currently running AI image.

Example:

Byte	**Data - Example**	**Description**
0..3	0x01 0x00 0x52 0x03	The version of the currently running firmware is 1.0.850.
4..5	0x01 0x02	The App interface version of the currently running firmware is 1.2.
6..7	0x02 0x00	The fitting and service interface version of the currently running firmware is 2.0.
8	0x04	The forced minimum version of the currently running firmware is 4.
9..12	0x01 0x02 0x03 0x04	The version of the currently running AI image is 1.2.1027.

4.6.5 Firmware upgrade status characteristic

Status of the firmware upgrade. For more details about the firmware upgrade protocol, see section 4.6.6.1.

Firmware upgrade status (R) – UUID b6f3df**07**-...			
Byte length	**Short description**	**Type**	**Comments**
1	Upgrade type	Uint8	Type of upgrade. 0: No upgrade started. 1: Firmware package upgrade. Other values may be used in the future to upgrade e.g., hearing aid data. When an upgrade is completed, it will be reset to 0. When Upgrade type is 0, all other fields in the characteristic will also be 0.
1	Upgrade version, Major	Uint8	Major version of the current firmware upgrade.
1	Upgrade version, Minor	Uint8	Minor version of the current firmware upgrade.
2	Upgrade version, Build	Uint16	Build version of the current firmware upgrade.
4	Upgrade offset	Uint32	Number of transferred bytes. This is also the byte offset for the next chunk of data to transfer.
4	Upgrade size	Uint32	Total size of the firmware upgrade.
1	Activation status	Uint8	Status of last firmware upgrade activation. Bit 0..6: 0: Firmware upgrade was not activated. 1: Firmware upgrade completed successfully. 2: Firmware upgrade activation was started but failed with an unknown error. 3: Firmware upgrade image verification failed. Bit 7:

Firmware upgrade status (R) – UUID b6f3df**07**-...			
			0: The activation request came from the Firmware Upgrade Transfer characteristic (i.e. App). 1: The activation request came from PI (e.g. fitting). The status is reset to 0 on reboot.

Example:

Byte	Data - Example	Description
0	0x01	A firmware package upgrade transfer has been started.
1..4	0x01 0x00 0x52 0x03	The version of the package being transferred is 1.0.850.
5..8	0xA0 0x86 0x01 0x00	100000 bytes of the firmware package have been transferred. The next transfer must have an offset of 100000.
9..12	0x90 0xD0 0x03 0x00	The total size of the firmware upgrade package is 250000 bytes.
13	0x00	No new firmware has been activated since the last reboot.

4.6.6 Firmware upgrade transfer characteristic

Characteristic to transfer data for the firmware upgrade.

Firmware upgrade transfer (R/W) – UUID b6f3df**08**-...			
Byte length	**Short description**	**Type**	**Comments**
4	Offset	Uint32	Must match the current “Upgrade offset” from Firmware upgrade status characteristic or be 0 to reset the upgrade.
Up to 508 bytes	Data	Blob	Chunk of data to transfer. When the entire package has been transferred (“Upgrade offset” = “Upgrade size”), sending the package signature as data will reset the HA and apply the new upgrade. The first chunk must contain the entire package header (315 bytes) and the last chunk, and the activation signature must have the exact size, but all other data chunks can have an arbitrary data length up to the max data size.

Example:

Byte	**Data - Example**	**Description**
0..3	0xA0 0x86 0x01 0x00	This chunk of data starts at byte offset 100000.
4..N	0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F 0x10 0x11 0x12 0x13 0x14 ...	The chunk of data being transferred.

4.6.6.1 Firmware upgrade protocol

To upgrade the HA firmware, the APP must first check if an upgrade with the firmware package is already started by reading the firmware upgrade status. If a firmware upgrade has been started with a different version, the APP can decide to restart the upgrade with its current version. It does not matter if the restarted upgrade version is earlier or later than the upgrade currently in progress.

To start a new upgrade or restart an upgrade, the APP sends the first chunk of the package to the HA using the transfer characteristic with an offset of zero. The first chunk must at least include the entire package header (315 bytes), or the HA will respond with an “Invalid attribute value length” error. The HA will then verify the header signature, and if that fails, it will respond with a “Write request rejected” error.

After that (or to continue an existing upgrade), the APP sends the next image chunk starting from the current upgrade offset. If the transfer offset does not match the current upgrade offset, the HA will respond with an “Attribute value not allowed” error and will ignore the chunk. The last chunk must contain the exact number of bytes remaining. Otherwise, the HA will respond with an “Invalid attribute value length” error and again ignore the chunk.

When the entire firmware package has been transferred (indicated in the status by the offset being equal to the size of the firmware package), the HA is ready to apply the new firmware. To do that, the APP sends the package signature using the transfer characteristic as a single transfer (with the offset still matching). If the signature matches, the HA will reboot into the new firmware. If not, the HA will respond with a “Write request rejected” error.

4.6.6.2 FW Upgrade related errors

This is an overview of the error codes from the firmware.

None of these should happen if the procedure is done correctly and the image is valid.

Please refer to [3] for further description on various error scenarios and how the FW handling these.

Error	Context/State	Error Code (GATT status code)	Reason
Incorrect header	Request FW upgrade	Write request rejected (0xFC)	FW Header signature verify failed
Incorrect header	Request FW upgrade	Invalid attribute value length (0x0D)	FW Header length mismatch
Incorrect FW upgrade offset provided	Upload FW data	Attribute value not allowed (0x13)	FW upgrade offset does not match with provided
Incorrect image	Upload FW data last chunk	Invalid attribute value length (0x0D)	The last chunk must contain the exact number of bytes remaining otherwise send error and ignore the chunk.
Incorrect signature	Requests activate	Invalid attribute value length (0x0D)	FW signature length is incorrect
Incorrect signature	Requests activate	Write request rejected (0xFC)	FW signature verification failed
Firmware upgrade is busy	Any	Procedure in progress (0xFE)	The firmware is currently handling a request from e.g. fitting

5 External interfaces

N/A

6 Constraints

N/A

7 Error handling

In some characteristics not all values are allowed. This is checked when a write is performed. The Firmware will validate and send proper ATT response if input is not allowed. If it is ok, then an ATT Success response is sent.

8 Terms and Abbreviations

Term	Description
N/A	N/A
N/A	N/A

Abbreviation	Description
API	Application Program Interface
BLE	Bluetooth Low Energy
HA	Hearing aid
GATT	Generic Attribute Profile

9 References

Reference	DOC ID	Details
1	https://www.bluetooth.com/specifications/specs/device-information-service-1-1/	Device Information Service 1.1
2	D00326700	Device Information Service for D-Chipset
3	D00293371	Firmware Upgrade Design Specification
4	D00286952	Persistent logs over BLE design specification