# HiQnet Third Party Programmers Quick-Start Guide

Every HiQnet message consists of a **HiQnet Header** and a **Payload**. All numeric fields are sent over the wire in "network-byte-order," ie. Big-Endian.

## HiQnet Header Structure

The structure of the header is as follows:

| | |
|---|---|
| VERSION | 1 byte |
| HEADER LENGTH | 1 byte |
| MESSAGE LENGTH | 4 bytes |
| SOURCE ADDRESS | 6 bytes |
| DESTINATION ADDRESS | 6 bytes |
| MESSAGE ID | 2 bytes |
| FLAGS | 2 bytes |
| HOP COUNT | 1 byte |
| SEQUENCE NUMBER | 2 bytes |
| Additional Variable Header | 0 or more bytes |

The **Version** Number indicates the revision number of the entire protocol; it is *not* used for differentiating between revisions of individual messages. The value is always "2".

The **Header Length** is the size in bytes of the entire message header, including any additional optional variable-length header.

The **Message Length** is the size in bytes of the entire message - from the 'Version' field through to the last byte of the payload.

The **Source Address** specifies where the message has come from.

The **Destination Address** specifies where the message is to be delivered.

The **Message ID** indicates the method that the destination node must perform. If there is a payload, it is usually specific to the type of method indicated by the Message ID.

The **Flags** denote what kinds of options are active when set to '1' and are allocated in the following manner:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Bit 15:<br>Reserved | Bit 14:<br>Reserved | Bit 13:<br>Reserved | Bit 12:<br>Reserved | Bit 11:<br>Reserved | Bit 10<br>Reserved | Bit 9:<br>Reserved | Bit 8:<br>Reserved |
| Bit 7:<br>Reserved | Bit 6:<br>Reserved) | Bit 5:<br>Guaranteed | Bit 4:<br>Reserved | Bit 3:<br>Error (header extension follows) | Bit 2:<br>Information | Bit 1:<br>Reserved | Bit 0:<br>Reserved |

**Hop Count** value is 5.

**Sequence Number** is for debugging. It may be zero, or its value may be incremented with every message sent starting from 1.

## HiQnet Header for the Discovery Message

| | | |
|---|---|---|
| VERSION | 1 byte | 2 |
| HEADER LENGTH | 1 byte | 25 |
| MESSAGE LENGTH | 4 bytes | 72 |
| SOURCE ADDRESS | 6 bytes | 1-65534 / 0<br>2 Bytes NODE / 4 Bytes VD-OBJECT |
| DESTINATION ADDRESS | 6 bytes | 1-65534 / 0<br>2 Bytes NODE / 4 Bytes VD-OBJECT |
| MESSAGE ID | 2 bytes | 0x0 |
| FLAGS | 2 bytes | 0x0020 (guaranteed bit set) |
| HOP COUNT | 1 byte | *5* |
| SEQUENCE NUMBER | 2 bytes | *0* |

The Message Length of 72 bytes includes the contents of the Discovery Payload Structure below.

### Discovery Payload Structure

| | |
|---|---|
| NODE | 2 bytes |
| COST | 1 byte |
| SERIAL NUMBER LENGTH | 2 bytes |
| SERIAL NUMBER | 16 bytes |
| MAX MESSAGE SIZE | 4 byte |
| KEEPALIVE PERIOD | 2 byte |
| NETWORK ID | 1 byte |
| MAC ADDRESS | 6 byte |
| DHCP | 1 byte |
| IP ADDRESS | 4 byte |
| SUBNET MASK | 4 byte |
| GATEWAY | 4 byte |

The **NODE** address is the source HiQnet address of the device or app. Its value is 1 through 65534. Zero and 65535 are reserved.

**COST** is always **1** when using Ethernet.

**SERIAL NUMBER LENGTH** is always “16”.

**SERIAL NUMBER** uniquely identifies the Node indicated by the Source Address. It is large enough to contain a UUID. Often, a device will have the first 10 bytes be zero and the last 6 bytes be the MAC address.

The **MAX MESSAGE SIZE** specifies the maximum size of a HiQnet message that the device (or app) is capable of receiving. Apps typically advertise 1,048,576 in this field.

**KEEPALIVE PERIOD** is in milliseconds. It is typically 10,000 (ten seconds).

**NETWORK ID** is always **1** for Ethernet.

**MAC ADDRESS** is the MAC address of the connecting client.

**DHCP** is set to **1** when the client is using a DHCP server. It is set to 0 to indicate static IP address.

**IP ADDRESS** is the IP address of the device or app, which will be used by the desk to connect back with this client upon receiving the discovery message.

**SUBNET MASK** is used to determine if devices and apps are in the same subnet.

**GATEWAY** is optional. Use 0.0.0.0 for "no gateway."

## Discovery Payload Example

<table>
<tr><td>NODE</td><td>2 bytes</td><td>1</td></tr>
<tr><td>COST</td><td>1 byte</td><td>1</td></tr>
<tr><td>SERIAL NO. LENGTH</td><td>2 bytes</td><td>16</td></tr>
<tr><td>SERIAL NUMBER</td><td>16 bytes</td><td>
<table>
<tr><td>0x00</td><td>0x00</td><td>0x17</td><td>0x24</td><td>0x82</td><td>0x3a</td><td>0xf2</td></tr>
<tr><td>Bytes<br>1 thru 10</td><td colspan="6">Bytes 11-16 (6-byte MAC Address)</td></tr>
</table>
</td></tr>
<tr><td>MAX MESSAGE SIZE</td><td>4 bytes</td><td>1048576 (1024*1024)</td></tr>
<tr><td>KEEP ALIVE PERIOD</td><td>2 bytes</td><td>10000</td></tr>
<tr><td>NETWORK ID</td><td>1 byte</td><td>1</td></tr>
<tr><td>MAC ADDRESS</td><td>6 bytes</td><td>0x00 0x17 0x24 0x82 0x3a 0xf2</td></tr>
<tr><td>DHCP</td><td>1 byte</td><td>1</td></tr>
<tr><td>IP ADDRESS</td><td>4 bytes</td><td>0x0a 0x01 0x13 0x0d (10.1.19.13)</td></tr>
<tr><td>SUBNET MASK</td><td>4 bytes</td><td>0xff 0xff 0x00 0x00 (255.255.0.0)</td></tr>
<tr><td>GATEWAY</td><td>4 bytes</td><td>0x00 0x00 0x00 0x00 (0.0.0.0)</td></tr>
</table>

## Connecting to the device from your app

The device will be listening on TCP port 3804. Create a TCP socket and connect to port 3804 on the device's IP address.

Send a Discovery message as the first message. The device will reply with a Discovery message with the **Info** bit set to 1 in the Flags. This lets you know that the device has recognized your app and can communicate.

## KeepAlive message

To keep the TCP connection alive, send the same Discovery Message periodically, with the flag's Info and Guaranteed bits set to 1. So the flags field would be 0x0024. The periodic time interval the device requested is advertised in its Disco message. It is typically ten seconds. Sending the Discovery Message every five seconds is always safe.

## Enabling Normal Session-less Communication

After establishing a connection with the client, some devices or apps will send a **HiQnet Hello Message** to your app. Message ID for this message is **0x08** and it is used to open a session between two HiQnet nodes. **Info** bit in the header of this message will not be set which means that this is a **Request** message and device is expecting a reply for this message. Your app should then reply with an error header to refuse sessions. Then normal session-less communication can occur.

## Example HiQnet Header for the Hello Session Request Message

| | | |
|---|---|---|
| VERSION | 1 byte | **2** |
| HEADER LENGTH | 1 byte | **25** |
| MESSAGE LENGTH | 4 bytes | **29** |
| SOURCE ADDRESS | 6 bytes | 1-65534 (2 Bytes NODE) / **0** (4 Bytes VD-OBJECT) |
| DESTINATION ADDRESS | 6 bytes | 1-65534 (2 Bytes NODE) / **0** (4 Bytes VD-OBJECT) |
| MESSAGE ID | 2 bytes | 0x8 |
| FLAGS | 2 bytes | 0 (or 0x20) |
| HOP COUNT | 1 byte | *5* |
| SEQUENCE NUMBER | 2 bytes | *0* |

### Payload for the Hello Message From Device

| | | |
|---|---|---|
| SESSION NUMBER | 2 bytes | 1-65535 |
| FLAG MASK | 2 bytes | 0x01FF |

**SESSION NUMBER** in the payload is a random number, which the device will be expecting in the HiQnet Headers unless the client refuses sessions with an Error Header response.

## Example HiQnet Error Header Response to Hello Message

| | | |
|---|---|---|
| VERSION | 1 byte | **2** |
| HEADER LENGTH | 1 byte | **29** |
| MESSAGE LENGTH | 4 bytes | **29** |
| SOURCE ADDRESS | 6 bytes | 1-65534 (2 Bytes NODE) / **0** (4 Bytes VD-OBJECT) |
| DESTINATION ADDRESS | 6 bytes | 1-65534 (2 Bytes NODE) / **0** (4 Bytes VD-OBJECT) |
| MESSAGE ID | 2 bytes | 0x8 |
| FLAGS | 2 bytes | 0x002c (Guar, Error, Info) |
| HOP COUNT | 1 byte | 5 |
| SEQUENCE NUMBER | 2 bytes | 0 |
| ERROR LENGTH | 2 bytes | 2 |
| ERROR MESSAGE | 2 bytes | 0x00 0x00 |

## Setting a Parameter on the Device

If a client wants to change some parameter on the device it has to send a MultiParamSet message, which has the Message ID of 0x0100.

## HiQnet Header for the MultiParamSet Message

| | | |
|---|---|---|
| VERSION | 1 byte | 2 |
| HEADER LENGTH | 1 byte | 25 |
| MESSAGE LENGTH | 4 bytes | 34 (depends on payload) |
| SOURCE ADDRESS | 6 bytes | 1-65534 / 0<br>2 Bytes NODE / 4 Bytes VD-OBJECT |
| DESTINATION ADDRESS | 6 bytes | 1-65534 / 0<br>2 Bytes NODE / 4 Bytes VD-OBJECT |
| MESSAGE ID | 2 bytes | 0x0100 |
| FLAGS | 2 bytes | 0x0020 (Guar) |
| HOP COUNT | 1 byte | 5 |
| SEQUENCE NUMBER | 2 bytes | 0 |

### Payload for the MultiParamSet Message

| | | |
|---|---|---|
| NUMBER OF Parameters | 2 bytes | 1 |
| PARAMETER ID | 2 bytes | 1 |
| DATATYPE | 1 byte | 4 |
| VALUE | 4 bytes | 1 |

**NUMBER OF PARAMETERS** can be greater than one if requesting more than one parameter in the same message. “PARAMETER ID” through “VALUE” will repeat “NUMBER OF PARAMETERS” times.

**PARAMETER ID** uniquely identifies the parameter within an Object.

**DATATYPE** is defined in the table below. In this example it was 4, which means LONG. How many bytes **VALUE** takes up depends on the **DATATYPE**.

| Name | 'C' Declaration | Range | Bytes | DATATYPE Enumeration |
|---|---|---|---|---|
| BYTE | Char | -128 to 127 | 1 | 0 |
| UBYTE | unsigned char | 0 to 255 | 1 | 1 |
| WORD | Short | -32768 to 32767 | 2 | 2 |
| UWORD | unsigned short | 0 to 65535 | 2 | 3 |
| LONG | long | -2,147,483,648 to 2,147,483,647 | 4 | 4 |
| ULONG | unsigned long | 0 to 4,294,967,926 | 4 | 5 |
| FLOAT32 | float | As IEEE-754 | 4 | 6 |
| FLOAT64 | double | As IEEE-754 | 8 | 7 |

**VALUE** takes up the number of bytes as determined by **TYPE**.

## Receiving a Parameter from the Device

The MultParamSet message is also used to receive Parameters from a Device**.** The Info bit in the flags field may or may not be set. For example, a Soundcraft device will not set the Info bit but a Crown device will. If an app is receiving MultiParamSet's from a Device, they contain the current values of Parameters in the Device. It does not mean the Device is trying to "set" parameters in the app, which would make no sense.

One way to get the Device to send a MultiParamSet to an app is for the app to send a MultiParamGet to the device.

## HiQnet Header for the MultiParamGet Message

| | | |
|---|---|---|
| VERSION | 1 byte | 2 |
| HEADER LENGTH | 1 byte | 25 |
| MESSAGE LENGTH | 4 bytes | 29 |
| SOURCE ADDRESS | 6 bytes | 1-65534 (2 Bytes NODE) \| 0 (4 Bytes VD-OBJECT) |
| DESTINATION ADDRESS | 6 bytes | 1-65534 (2 Bytes NODE) \| 0 (4 Bytes VD-OBJECT) |
| MESSAGE ID | 2 bytes | 0x103 |
| FLAGS | 2 bytes | 0x0020 (Guar) |
| HOP COUNT | 1 byte | 5 |
| SEQUENCE NUMBER | 2 bytes | 0 |

### Payload for the MultiParamGet Message

| | | |
|---|---|---|
| PARAMETER COUNT | 2 bytes | 1 |
| PARAMETER ID | 2 bytes | 1 |

For the above example the MultiParamGet is asking for one Parameter from the device, parameter ID 1 living on Virtual Device 0, Object 0.

**PARAMETER COUNT** is the number of parameters that live in this same object that are being requested.

**PARAMETER ID** will repeat "PARAMETER COUNT" times.

The response to a MultiParamGet is a MultiParamSet as mentioned above. If an error header comes back instead, then the parameter does not exist on the device.

## Subscribing to a value change on the device

The other way to get the device to send MultiParamSet messages is for the app to send the ParamSubscribeAll to the device. The message ID is 0x113.

Suppose the product XML for Soundcraft's Si Expression 2 is something like shown below. Then subscribing to the Object with Object Address 2 will notify the client of changes to any of the four state variables in the Object: the On Button states and the Fader level values for LR and Mono.

Any time one of these parameters change, the device will immediately send a MultiParamSet to the app with the new current value.

```
<?xml version="1.0"?>
<Product ClassID="909" ClassName="Si Expression 2">
  <VirtualDevices>
    <VirtualDevice Index="1" Name="Public VD Si Expression 2">
      <Objects>
        <Object ClassID="402" Name="Main masters" Address="2">
          <StateVars>
            <StateVar ClassID="2404" Name="Main LR Master On" DataType="4" ID="1" />
            <StateVar ClassID="2404" Name="Main Mono Master On" DataType="4" ID="2" />
            <StateVar ClassID="2403" Name="Main LR Master Level" DataType="4" ID="3" />
            <StateVar ClassID="2403" Name="Main Mono Master Level" DataType="4" ID="4" />
          </StateVars>
        </Object>
</Objects>
    </VirtualDevice>
  </VirtualDevices>
</Product>
```

## HiQnet Header for the SubscribeAll Message From Client

| | | |
|---|---|---|
| VERSION | 1 byte | 2 |
| HEADER LENGTH | 1 byte | 25 |
| MESSAGE LENGTH | 4 bytes | 36 |
| SOURCE ADDRESS | 6 bytes | 1-65534 (2 Bytes NODE) / 0 (4 Bytes VD-OBJECT) |
| DESTINATION ADDRESS | 6 bytes | 1-65534 (2 Bytes NODE) / **0x01000002** (4 Bytes VD-OBJECT) |
| MESSAGE ID | 2 bytes | 0x113 |
| FLAGS | 2 bytes | 0x0020 (Guar) |
| HOP COUNT | 1 byte | 5 |
| SEQUENCE NUMBER | 2 bytes | 0 |

How is the value of VD-OBJECT calculated as **0x01000002 (decimal 16777218)?**

VD-OBJECT is a 4 byte value which has 1st (most significant) byte corresponding to the Virtual Device and remaining (least significant) 3 bytes corresponding to the Object address:

| VD, 1 byte | Object, 3 bytes |
|---|---|
| 0x01 | 0x000002 |

…as per the example in the XML above, where the Virtual Device index is 1 and the Object address is 2.

The binary representation of above 4 bytes is
0000 0001 0000 0000 0000 0000 0000 0010
which is equal to **16777218** in decimal.

### Payload for the SubscribeAll Message From Client

| | | |
|---|---|---|
| NODE ID | 2 bytes | 1-65534 |
| VD-OBJECT | 4 bytes | 0x01000002 |
| Change Type | 1 byte | 5 |
| Sensor Rate | 2 bytes | 50 |
| Initial Update | 2 bytes | 1 |

**Change Type** is a bit field for newer devices, and an enumerated number for older devices. Its purpose is to tell the device which kind of parameters the client app wants to subscribe to.

For older devices, the value of ChangeType is:

```
enum ChangeType
{
   AllParametersAndAttributes = 0,
   NonSensorParametersAndAttributes = 1,
   SensorParameters = 2
}
```

For newer devices, the value of ChangeType is:

Bit field:
Bit 0: Non-Sensor SVs
Bit 1: Sensor SVs
Bit 2: Attributes
Bit 3-7: not used (zero)

The value for Soundcraft consoles is always **5,** which means we are subscribing to Non-Sensor SV's and Attributes (bits zero and two are set).

Older devices include those by:

- AKG
- dbx
- JBL
- Some older Crown amplifiers

Newer devices include those by:

- BSS Audio
- Newer Crown amplifiers
- Soundcraft

When in doubt, try the bit-field version of ChangeType first.

**Sensor Rate** is the value in milliseconds which is sent to get the updates of the parameters at a periodic interval. 50 is a typical value. Sensor parameters are things that change periodically, such as meters and LED's. Soundcraft consoles do not have sensor parameters but all other devices do.

**Initial Update** is zero or one. Zero means just set up the subscription. One means set up the subscription and immediately send all the current values of the subscribed parameters. One is the typical value here.

## MultiParamSubscribe

MultiParamSubscribe, Message ID 0x010F, is used to subscribe to individual parameters rather than whole objects. The example below shows subscribing to just a couple of the parameters in the XML example above. We will subscribe to just the "Master Level" parameters.

| | | |
|---|---|---|
| VERSION | 1 byte | 2 |
| HEADER LENGTH | 1 byte | 25 |
| MESSAGE LENGTH | 4 bytes | 59 |
| SOURCE ADDRESS | 6 bytes | 1-65534 (2 Bytes NODE) \| 0 (4 Bytes VD-OBJECT) |
| DESTINATION ADDRESS | 6 bytes | 1-65534 (2 Bytes NODE) \| **0x01000002** (4 Bytes VD-OBJECT) |
| MESSAGE ID | 2 bytes | 0x010F |
| FLAGS | 2 bytes | 0x0020 |
| HOP COUNT | 1 byte | 5 |
| SEQUENCE NUMBER | 2 bytes | 0 |
| **Payload…** | | |
| No of Subscriptions | 2 bytes | 2 |
| Publisher Param ID | 2 bytes | 3 |
| Subscription Type | 1 byte | 0 |
| Subscriber Address | 6 bytes | (same as Source Address) |
| Subscriber Param ID | 2 bytes | 3 |
| Reserved | 1 byte | 0 – Reserved |
| Reserved | 2 bytes | 0 – Reserved |
| Sensor Rate | 2 bytes | 50 (ignored for non-sensors) |
| Publisher Param ID | 2 bytes | 4 |
| Subscription Type | 1 byte | 0 |
| Subscriber Address | 6 bytes | (same as Source Address) |
| Subscriber Param ID | 2 bytes | 4 |
| Reserved | 1 byte | 0 – Reserved |
| Reserved | 2 bytes | 0 - Reserved |
| Sensor Rate | 2 bytes | 50 (ignored for non-sensors) |

**No of Subscriptions** is the number of parameters that will follow. **Publisher Param ID** through **Sensor Rate** will repeat **No of Subscriptions** times.

**Publisher Param ID** is the parameter ID to subscribe. In this example we are referring to "Main LR Master Level", ID=3, and in the second subscription, "Main Mono Master Level", ID=4.

**Subscription Type** is always zero.

**Subscriber Address** is the same as the Source Address.

**Subscriber Param ID** is the same as the Publisher Param ID.

**Sensor Rate** is for meters or LEDs and is typically 50 milliseconds. The MultiParamSubscribe message is a good way to subscribe to just the individual sensor parameters you want.

## Receiving Subscription Data at the Client

Whether you use SubscribeAll or MultiParamSubscribe, data will come back from the device in the form of MultiParamSet's. Non-sensor parameters, such as faders, will arrive on the same TCP connection used for the original subscribe message. Sensor parameters, such as meters, will arrive on UDP port 3804 as UDP frames. So if you subscribe to any sensor SV's, you will need to bind a UDP socket to port 3804 and "ReceiveFrom" the UDP socket to get the MultiParamSet's.

## Goodbye

When a client wants to disconnect with the desk it has to send a **Goodbye** message to the desk so that the desk will close its current session with the client. The message ID associated with this Goodbye message is **0x7. Goodbye** cancels all subscriptions. Close the TCP connection after sending **Goodbye**. There is no response to Goodbye. The device will **NOT** send a Goodbye with the Info bit set in the flags field.

## HiQnet Header for the Goodbye Message From Client

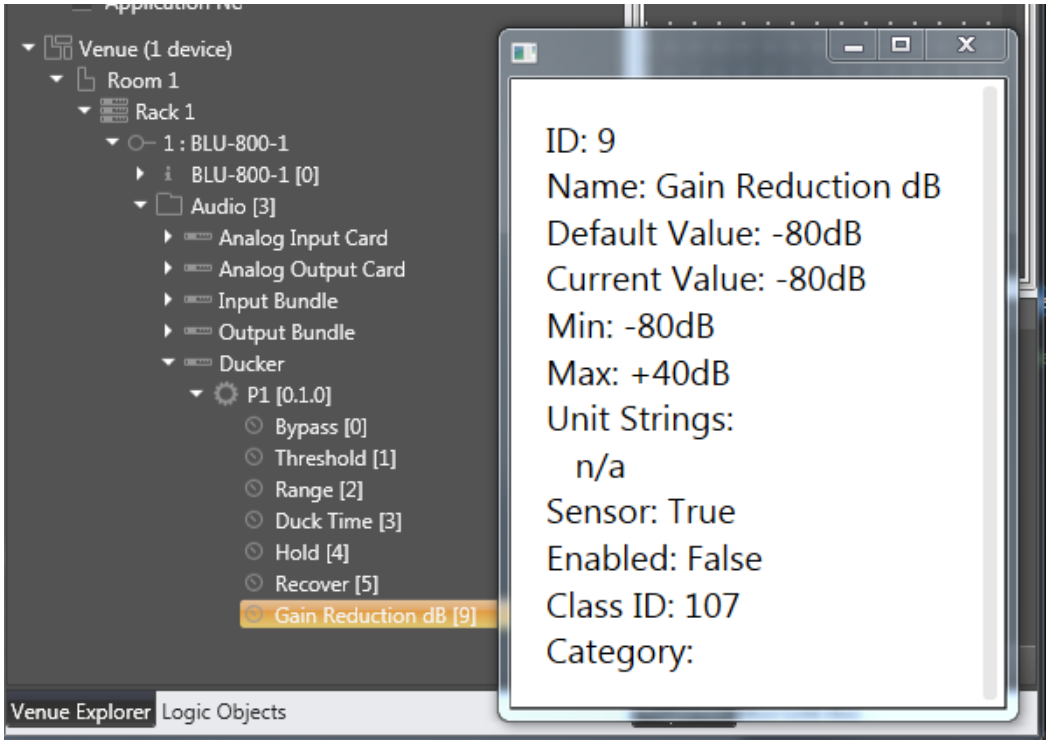
| | | |
|---|---|---|
| VERSION | 1 BYTE | 2 |
| HEADER LENGTH | 1 BYTE | 25 |
| MESSAGE LENGTH | 4 BYTE | 27 |
| SOURCE ADDRESS | 6 BYTE | 1-65534 / 0<br>2 Bytes NODE / 4 Bytes VD-OBJECT |
| DESTINATION ADDRESS | 6 BYTE | 1-65534 / 0<br>2 Bytes NODE / 4 Bytes VD-OBJECT |
| MESSAGE ID | 2 BYTE | 0x7 |
| FLAGS | 2 BYTE | 0x0020 |
| HOP COUNT | 1 BYTE | 5 |
| SEQUENCE NUMBER | 2 BYTE | 0 |

### Payload for the Goodbye Message From Client

| | | |
|---|---|---|
| NODE ID | 2 BYTE | 12345 |

## Device Objects and Parameters

Devices have an Object, Parameter, and Attribute tree structure. Run Audio Architect or System Architect and view the tree structure of a device by dragging an offline device of the type you are interested in onto the Venue and using the explorer. For example:



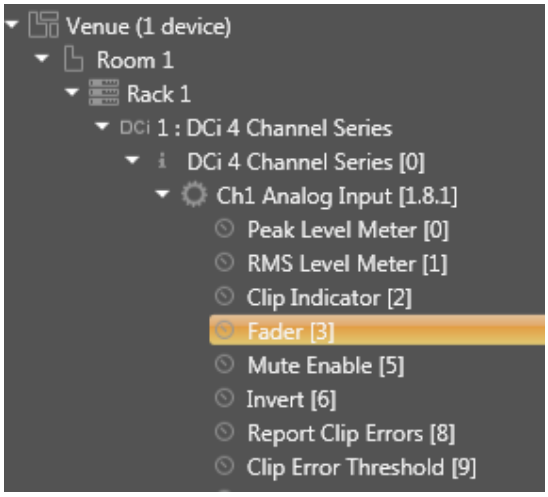

By expanding the Venue Explorer, we can see that if we want to subscribe to the "Gain Reduction dB" parameter, we would use a **DESTINATION ADDRESS** of 0x0001 for the "Node" (or whatever the node address is for this device) and 0x03000100 for the VD-OBJECT portion. That's because the "Audio" Virtual Device is at ID [3], and the "P1" Object is at [0.1.0]. Then, for the **Publisher Param ID** we would use 9, since "Gain Reduction" is Parameter ID [9].

By double-clicking on the "Gain Reduction" parameter, we get a window describing the Attributes that belong to this parameter. Attributes are metadata that have names and values. The "Min" attribute is -80db and the "Max" attribute is +40db, which means that the "Gain Reduction" parameter is guaranteed to have a value somewhere in between -80db and +40db (inclusive).

Note, the "Gain Reduction" parameter is a meter (Sensor=True), so MultiParamSet's from the device will arrive periodically (every 50ms) on UDP port 3804 at the client app.

## MultiParamSubscribe example

For a 4-channel Crown DCi Series amplifier, say the app subscribes to the Channel 1 Analog Input Fader.




We see the HiQnet Node Address is 1, the Virtual Device is 0, the Ch1 Analog Input Object Address is 1.8.1, and the Fader Parameter ID is 3. So a MultiSVSubscribe from our app at HiQnet Node Address 51 would look like this:

02 19 00 00 00 2B 00 33 00 00 00 00 00 01 00 01 08 01 01 0F 00 20 05 00 00 00 01 00 03 00 00 01 00 01 08 01 00 03 00 00 00 00 00

A fast way to generate strings like this is to use Audio Architect's Third-party Controller tool. You can drag the parameters you want onto the form, and generate HiQnet message strings for Set, Subscribe, and Unsubscribe.

For more information and more messages, see the HiQnet *Third Party Programmer's Guide* and the *Guide to Soundcraft's HiQnet Connectivity* documents.