Common System Exclusive Commands Manual

1. Overview

Header:

0xF0 - start of system exclusive
0x00, 0x01, 0x73 - iConnectivity's manufacturer ID code
0x7E - message class

Body:

Device ID - 7 bytes (2 bytes for product ID, 5 bytes for serial number)
Transaction ID - 2 bytes (0 - 16383)
Command - 2 bytes
Data Length - 2 bytes, number of bytes in Command Data (0 - 16383)
Command Data - length varies depending on command
Checksum - 1 byte

Footer:

0xF7 - end of system exclusive

Data Format

All fields are big-endian (most significant byte occurs first, least significant byte occurs last). All fields which accept values greater than 0x7F must have the value split across multiple bytes in order to be compatible with MIDI sysex format. The least significant 7 bits of the value are contained in the 7 least significant bits of the last byte. The next 7 bits are contained in the 7 least significant bits of the last-1 byte, etc. Here are some examples:

Value	First Byte	Second Byte
0x0003	0x00	0x03
0x007F	0x00	0x7F
0x0080	0x01	0x00
0x0081	0x01	0x01
0x1234	0x24	0x34
0x2CA5	0x59	0x25

Device Identifier

The first 2 bytes of Device ID are product ID (PID) a 14 bit value split across 2 bytes. The remaining 5 bytes are serial number (SNUM) a 32 bit value split across 5 bytes. PID and SNUM must be formatted so that each byte is 0x7F or less as explained in the Data Format section. Here are some examples:

PID	SNUM	Device Identifier
0x0001	0x00000001	0x00 0x01 0x00 0x00 0x00 0x00 0x01
0x0001	0x00000080	0x00 0x01 0x00 0x00 0x00 0x01 0x00
0x00A0	0x00000080	0x01 0x02 0x00 0x00 0x00 0x01 0x00
0x0ABC	0x12345678	0x15 0x3C 0x01 0x11 0x51 0x2C 0x78

Some commands allow PID and SNUM to be zero which indicates a wildcard as follows:

PID	SNUM	Meaning
= 0	= 0	All devices.
= 0	> 0	All devices with the specified serial number.
> 0	= 0	All devices with the specified product ID.
> 0	> 0	A device that matches the specified product ID and serial number.

Transaction ID

Transaction ID can be used to uniquely identify messages. Typically, the host sends a query message to one or more devices, each device then sends a response message back to the host. A device will return the same transaction ID in the response as was sent in the query. It is not required that transaction IDs increment for every message sent by a host, they can always be 0x0000 or some other value, it is up to the host whether or not it wants to use transaction ID for managing messages. Transaction ID is a 14 bit value split across 2 bytes. Each byte is 0x7F or less as explained in the Data Format section. Range is 0 - 16383.

Command

A 14 bit field split across 2 bytes. The least significant 10 bits are the command ID. The most significant 4 bits are command flags:

Bit	Description
13	0 = Answer/Read, 1 = Query/Write
12-10	reserved (always zero)

Bit	Description
9-0	command ID

Data Length

The number of bytes that follow in the command data section. Data length is a 14 bit value split across 2 bytes. Each byte is 0x7F or less as explained in the Data Format section. Range is 0 - 16383.

Command Data

The content and length of the this field depends on the command ID. Some commands may have no command data.

Checksum

This field contains the 2s complement of the sum of all bytes in the body, excluding the checksum byte. In other words, summing all the bytes in the body should result in 0x00. Note that the checksum byte must be 0x7F or less to be compatible with MIDI sysex format (i.e. if the calculated checksum value is 0x83 then the checksum value used in the sysex message should be 0x03).

2. Device Commands

The following commands are defined for protocol version = 1.

2.1. GetDevice (Command ID = 0x01)

This command is used to discover devices. The sender can specify a specific product ID (PID), a specific serial number (SNUM), both PID & SNUM, or neither (using wildcards). This command should always be the first command sent to a device because the response contains useful information regarding maximum packet size that can be used for other commands.

Command Data

Command flags are Query.
Command ID is 0x01.
Data length is 0.

Example: discover all devices, any product ID, any serial number

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x00                      // product ID (all)
0x00, 0x00, 0x00, 0x00, 0x00    // serial number (all)
0x00, 0x00                      // transaction ID
0x40, 0x01                      // command flags and ID
0x00, 0x00                      // data length
```

```
0x3F                          // checksum
0xF7                          // footer
```

Example: discover all devices that are a specific product ID (3), any serial number

```
0xF0, 0x00, 0x01, 0x73, 0x7E  // header
0x00, 0x03                    // product ID (3)
0x00, 0x00, 0x00, 0x00, 0x00  // serial number (all)
0x00, 0x00                    // transaction ID
0x40, 0x01                    // command flags and ID
0x00, 0x00                    // data length
0x3C                          // checksum
0xF7                          // footer
```

2.2. RetDevice (Command ID = 0x02)

This command is sent by devices in response to a GetDevice command. It contains basic information that a host will need to use to communicate with this device (e.g. protocol version number). A host should cache this information and use it for all further communication with this device.

Command Data

Command flags are Answer.
Command ID is 0x02.
Data length depends on protocol version. For protocol version = 1, data length is always 4.

Byte #1: protocol version number that this device supports (1 - 127). If the host does not understand the protocol version number then it should not attempt any further communication with the device.

For protocol version number = 1:

Byte #2: device's current operating mode:

Value	Description
1	application mode
2	boot loader mode
3	test mode

Bytes #3-4: maximum length allowed for command data field (0 - 16383). If the host needs to send a message to the device that is longer than the allowed maximum length the host will need to split the command into multiple messages.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E  // header
```

```
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x02                      // command flags and ID
0x00, 0x04                      // data length
0x01                            // protocol version number (1)
0x01                            // application mode (1)
0x02, 0x00                      // maximum length of command data field (256)
0xxx                            // checksum
0xF7                            // footer
```

2.3. GetCommandList (Command ID = 0x03)

This command is used to query a device about which command IDs it supports. Devices that support this command will respond with a RetDevice message.

Command Data

Command flags are Query.
Command ID is 0x03.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x03                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

2.4. RetCommandList (Command ID = 0x04)

This command is sent by devices in response to a GetCommandList command. It contains a list of the command IDs that this device supports (command IDs that a host can send to this device to get info or change settings, not command IDs that this device will return to a host). Note that command IDs 0x01 and 0x03 are never returned (it is assumed that all devices support these command IDs).

Command Data

Command flags are Answer.
Command ID is 0x04.
Data length is 2 x the number of command IDs returned.

Bytes #1-2: command ID #1.
Bytes #3-4: command ID #2.
etc.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x03                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x00, 0x04                     // command flags and ID
0x00, 0x02                     // data length
0x00, 0x05                     // command ID #1 (GetInfoList)
0xxx                           // checksum
0xF7                           // footer
```

2.5. GetInfoList (Command ID = 0x05)

This command is used to query a device about which info IDs it supports. Devices that support this command will respond with a RetInfoList message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x05.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x03                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x40, 0x05                     // command flags and ID
0x00, 0x00                     // data length
0xxx                           // checksum
0xF7                           // footer
```

2.6. RetInfoList (Command ID = 0x06)

This command is sent by devices in response to a GetInfoList command. It contains a list of the info IDs that this device supports.

Command Data

Command flags are Answer.
Command ID is 0x06.
Data length is 2 x the number of info IDs returned.

Byte #1: info ID #1.
Byte #2: maximum length of info ID #1 (0 if read only).
Byte #3: info ID #2.
Byte #4: maximum length of info ID #2 (0 if read only).
etc.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x03                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x00, 0x06                     // command flags and ID
0x00, 0x06                     // data length
0x01, 0x00                     // info field ID #1 (accessory name, read only)
0x05, 0x00                     // info field ID #2 (firmware version, read only)
0x10, 0x1F                     // info field ID #3 (device name, 31 characters max)
0xxx                           // checksum
0xF7                           // footer
```

2.7. GetInfo (Command ID = 0x07)

This command is used to query a device about a single info ID. Devices that support this command will respond with a RetInfo message. If this command or the info ID is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x07.
Data length is 1.

Example: get firmware version

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x03                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x40, 0x07                     // command flags and ID
0x00, 0x01                     // data length
0x05                           // info ID for firmware version
0xxx                           // checksum
0xF7                           // footer
```

2.8. RetInfo / SetInfo (Command ID = 0x08)

RetInfo is sent by devices in response to a GetInfo command. It returns the ASCII string for a specific info ID.

SetInfo is used to set the ASCII string for a specific info ID. Devices that support this command will respond with an ACK message. Not all info IDs can be set, many are read only.

Command Data

Command flags are Answer for RetInfo, Write for SetInfo.
Command ID is 0x08.
Data length is 1 + the length of the info string.

Byte #1: info ID

Info ID	Description
0x01	accessory name (read only)
0x02	manufacturer name (read only)
0x03	model number (read only)
0x04	serial number (read only)
0x05	firmware version (read only)
0x06	hardware version (read only)
0x10	device name (read/write). Length must be at least two characters. First character must be a letter (A-Z, a-z). Only letters (A-Z, a-z), numbers (0-9) and some special characters (space, underscore, period, comma, minus, plus, forward slash, round brackets, angle brackets, square brackets, curly brackets) allowed.

Bytes #2 - #N: 7-bit ASCII string, not NULL terminated (length varies)

Example: return firmware version

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x08                      // command flags and ID
0x00, 0x06                      // data length
0x05                            // info ID (firmware version)
0x31, 0x2E, 0x30, 0x2E, 0x37    // info value (the ascii string “1.0.7”)
0xxx                            // checksum
0xF7                            // footer
```

Example: set device name to “MIDI1”

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x08                      // command flags and ID
0x00, 0x06                      // data length
0x10                            // info ID (device name)
0x4D, 0x49, 0x44, 0x49, 0x31    // info value (the ascii string “MIDI1”)
0xxx                            // checksum
0xF7                            // footer
```


2.9. GetResetList (Command ID = 0x09)

This command is used to query a device about which reset IDs it supports. Devices that support this command will respond with a RetResetList message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x09.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x09                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

2.10. RetResetList (Command ID = 0x0A)

This command is sent by devices in response to a GetResetList command. It contains a list of the reset IDs that this device supports.

Command Data

Command flags are Answer.
Command ID is 0x0A.
Data length is the number of reset IDs returned.

Byte #1: reset ID #1.
Byte #2: reset ID #2.
etc.

Reset ID	Description
0x01	restart into application mode
0x02	restart into boot loader mode

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
```

```
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x0A                      // command flags and ID
0x00, 0x02                      // data length
0x01                            // reset ID #1 (restart into application mode)
0x02                            // reset ID #2 (restart into boot loader mode)
0xxx                            // checksum
0xF7                            // footer
```

2.11. GetSaveRestoreList (Command ID = 0x0B)

This command is used to query a device about which save/restore IDs it supports. Devices that support this command will respond with a RetSaveRestoreList message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x0B.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x0B                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

2.12. RetSaveRestoreList (Command ID = 0x0C)

This command is sent by devices in response to a GetSaveRestoreList command. It contains a list of the save/restore IDs that this device supports.

Command Data

Command flags are Answer.
Command ID is 0x0C.
Data length is the number of save/restore IDs returned.

Byte #1: save/restore ID #1.
Byte #2: save/restore ID #2.
etc.

Save/Restore ID	Description
0x01	Save current settings to FLASH/EEPROM: The device will use these settings the next time it is restarted.
0x02	Restore user settings from FLASH//EEPROM: The current settings are discarded and the device is reset. If user settings were previously stored in FLASH/EEPROM then they are used, otherwise the factory default settings are used.
0x03	Restore factory default settings: Any previously saved settings are deleted from FLASH/EEPROM. The device is then reset and the factory default settings are used.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x0C                      // command flags and ID
0x00, 0x03                      // data length
0x01                            // reset ID #1 (save current settings to FLASH/EEPROM)
0x02                            // reset ID #2 (restore settings from FLASH/EEPROM)
0x03                            // reset ID #3 (restore settings to factory default)
0xxx                            // checksum
0xF7                            // footer
```

2.13. GetEthernetPortInfo (Command ID = 0x0D)

This command is used to query a device about a specific ethernet port/jack. Devices that support this command will respond with a RetEthernetPortInfo message. If this command is not supported the device will respond with an ACK message (with error code). The host should issue a GetMIDIInfo message to the device to discover the number of ethernet ports/jacks before issuing this command.

Command Data

Command flags are Query.
Command ID is 0x0D.
Data length is 2.

Bytes #1-2: ethernet port/jack ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x0D                      // command flags and ID
```

```
0x00, 0x02          // data length
0x00, 0x01          // ethernet port/jack ID
0xxx                // checksum
0xF7                // footer
```

2.14. RetEthernetPortInfo / SetEthernetPortInfo (Command ID = 0x0E)

RetEthernetPortInfo is sent by devices in response to a GetEthernetPortInfo command.

SetEthernetPortInfo is sent by a host to a device to set the values of some of the port/jack parameters. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored. The host only needs to send up to (and including) byte #19, all bytes beyond #19 are read-only and are ignored.

Command Data

Command flags are Answer for RetEthernetPortInfo, Write for SetEthernetPortInfo.
Command ID is 0x0E.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: ethernet port/jack ID (1 - N).

Byte #4 [W]: IPMode, 0 = use static IP (as defined in bytes 5 - 19), 1 = use dynamic IP (DHCP or AutoIP if DHCP server is not available).

Bytes #5 - 19 [W]: Device IP address, subnet mask, and gateway address to use when IPMode = static. All are four-byte big-endian values encoded in five-bytes (similar to device ID) so that the most significant bit of each byte is 0.

Byte	Description
5 - 9 [W]	The IP address to use when IPMode = static
10 - 14 [W]	The subnet mask to use when IPMode = static
15 - 19 [W]	The address of the gateway to use when IPMode = static

Bytes #20 - 34: Current (active) device IP address, subnet mask, and gateway address. All are four-byte big-endian values encoded in five-bytes (similar to device ID) so that the most significant bit of each byte is 0. These values are only valid if the ethernet port is connected to a network, is active, and (if IPMode = dynamic) has obtained values from a DHCP server or AutoIP has finished negotiation.

Byte	Description
20 - 24	Current device IP address
25 - 29	Current subnet mask
30 - 34	Current gateway address

Bytes #35 - 46: Ethernet MAC address for this port, 48-bit value encoded as a 12 character, 7-bit ASCII string, not NULL terminated.
Byte #47: Length of device name (Bonjour name) field that follows (0 - N).
Bytes #48-N: Device name (Bonjour name), 7-bit ASCII string, not NULL terminated. Device name is only valid if the ethernet port is connected to a network, is active, and has finished Bonjour name resolution.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x0E                      // command flags and ID
0x00, 0x31                      // data length (51)
0x01                            // command version number (1)
0x00, 0x01                      // ethernet port/jack ID
0x01                            // IPMode: dynamic
0x0C, 0x05, 0x20, 0x02, 0x64    // static IP address (192.168.1.100)
0x0F, 0x7F, 0x7F, 0x7E, 0x00    // static subnet mask (255.255.255.0)
0x0C, 0x05, 0x20, 0x02, 0x01    // static gateway address (192.168.1.1)
0x0A, 0x4F, 0x78, 0x00, 0x08    // current IP address (169.254.0.8)
0x0F, 0x7F, 0x7C, 0x00, 0x00    // current subnet mask (255.255.0.0)
0x0A, 0x4F, 0x78, 0x00, 0x01    // current gateway address (169.254.0.1)
0x41, 0x43, 0x37, 0x41,         // ethernet MAC address as ASCII string “AC7A42010203”
0x34, 0x32, 0x30, 0x31,
0x30, 0x32, 0x30, 0x32
0x04                            // length of device name (Bonjour name) field that follows
0x69, 0x43, 0x4D, 0x34          // the ASCII string “iCM4”
0xxx                            // checksum
0xF7                            // footer
```

2.15. ACK (Command ID = 0x0F)

This command is sent by devices in response to various commands (acknowledgement that a command succeeded or failed).

Command Data

Command flags are Answer.
Command ID is 0x0F.
Data length is 3.

Bytes #1-2: command that generated this ACK

Byte #3: error code

Error Code	Description
0x00	no error
0x01	unknown command
0x02	malformed message
0x03	command failed

Example: no error response to SetInfo command

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x0F                      // command flags and ID
0x00, 0x03                      // data length
0x40, 0x08                      // command that generated this ACK (SetInfo)
0x00                            // error code (no error)
0xxx                            // checksum
0xF7                            // footer
```

2.16. Reset (Command ID = 0x10)

This command is used to reset a device. Devices that support this command will respond with an ACK message and then reset itself after a short delay (usually around 1 second). If this command or the reset ID is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Write.
Command ID is 0x10.
Data length is 1.

Example: reset device so that it restarts into boot loader mode

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x10                      // command flags and ID
0x00, 0x01                      // data length
0x02                            // reset ID for restart into boot loader mode
```



```
0xxx                        // checksum
0xF7                        // footer
```

2.17. SaveRestore (Command ID = 0x11)

This command is used to save or restore the current device settings. Devices that support this command will respond with an ACK message. If this command or the save/restore ID is not supported the device will respond with an ACK message (with error code).

Note that restoring settings from FLASH/EEPROM or restoring settings to the factory default results in the device being reset.

Command Data

Command flags are Write.
Command ID is 0x11.
Data length is 1.

Example: restore to factory default

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x11                      // command flags and ID
0x00, 0x01                      // data length
0x03                            // save/restore ID for restore current to factory default
0xxx                            // checksum
0xF7                            // footer
```

2.18. GetGizmoCount (Command ID = 0x12)

This command is used to query a device about the number of other devices (gizmos) that are connected to it. Devices should previously have been sent a GetDevice command before issuing this command. Devices that support this command will respond with a RetGizmoCount message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x12.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x12                      // command flags and ID
0x00, 0x00                      // data length
```

```
0xxx                          // checksum
0xF7                          // footer
```

2.19. RetGizmoCount (Command ID = 0x13)

This command is sent by devices in response to a GetGizmoCount command. It contains the number of devices (gizmos) that are connected to this device.

Command Data

Command flags are Answer.
Command ID is 0x13.
Data length is always 2.

Bytes #1-2: gizmo count

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E  // header
0x00, 0x03                    // product ID
0x01, 0x02, 0x03, 0x04, 0x05  // serial number
0x00, 0x00                    // transaction ID
0x00, 0x13                    // command flags and ID
0x00, 0x02                    // data length
0x00, 0x04                    // gizmo count (4)
0xxx                          // checksum
0xF7                          // footer
```

2.20. GetGizmoInfo (Command ID = 0x14)

This command is used to query a device about a specific gizmo. Devices should previously have been sent a GetDevice and GetGizmoCount command before issuing this command. Devices that support this command will respond with a RetGizmoInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x14.
Data length is 2.

Bytes #1-2: gizmo ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E  // header
0x00, 0x05                    // product ID
0x01, 0x02, 0x03, 0x04, 0x05  // serial number
0x00, 0x00                    // transaction ID
0x40, 0x14                    // command flags and ID
0x00, 0x02                    // data length
```

```
0x00, 0x01          // gizmo ID
0xxx                // checksum
0xF7                // footer
```

2.21. RetGizmoInfo (Command ID = 0x15)

This command is sent by devices in response to a GetGizmoInfo command. It contains information for a specific gizmo.

Command Data

Command flags are Answer.
Command ID is 0x15.
Data length depends on command version number. For version = 1, data length is always 13.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte	Description
2 - 3	gizmo ID (1 - N)
4	gizmo type code: 1 = Query / Write (source), 2 = Answer / Read (destination)
5 - 6	port ID as used in MIDI commands (1 - N)
7 - 13	iConnectivity device identifier, all zeros for non-iConnectivity gizmos

The gizmo type code can be used to determine if a gizmo is upstream (source) or downstream (destination) of the device that is sent the GetGizmoInfo command. Sources generate queries, destinations provide answers.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x15                      // command flags and ID
0x00, 0x0D                      // data length
0x01                            // command version number (1)
0x00, 0x01                      // gizmo ID (1)
0x01                            // gizmo type (destination)
0x00, 0x02                      // port ID (2)
0x00, 0x05                      // product ID of gizmo
0x05, 0x04, 0x03, 0x02, 0x01    // serial number of gizmo
0xxx                            // checksum
```

0xF7 // footer

2.22. GetDeviceMode (Command ID = 0x16)

This command is used to query a device about the different operating modes that the device supports. Devices that support this command will respond with a RetDeviceMode message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x16.
Data length is 0.

Example:

0xF0, 0x00, 0x01, 0x73, 0x7E // header
0x00, 0x05 // product ID
0x01, 0x02, 0x03, 0x04, 0x05 // serial number
0x00, 0x00 // transaction ID
0x40, 0x16 // command flags and ID
0x00, 0x00 // data length
0xxx // checksum
0xF7 // footer

2.23. RetDeviceMode / SetDeviceMode (Command ID = 0x17)

RetDeviceMode is sent by devices in response to a GetDeviceMode command.

SetDeviceMode is sent by a host to a device to configure the operating mode. Data is organized into blocks. Some blocks may be read-only, others may have read-write parameters. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetDeviceMode, Write for SetDeviceMode.
Command ID is 0x17.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2: number of device mode blocks that follow (0 – N).
Bytes #3-N: device mode blocks:

Device Mode Block:

Byte	Description
1	Size of this block (including this byte).
2	Device mode block type: 1: sysex support 2: chain route map
3 - N	Dependent on device feature type (see tables below).

For device mode block type = 1 (sysex support):

Byte	Description
3 [W]	Current sysex mode. SetDeviceMode message only needs to send up to and including this byte when changing the current sysex mode; the remaining bytes in this block are read-only and can be skipped.
4	Number of sysex modes supported by this device (number of bytes that follow).
5 - N	Sysex modes: 1: Network Mode 2: Chain Mode

For device mode block type = 2 (chain route map), only used if chain mode is supported:

Byte	Description
3 [W]	Number of bytes in bitmask field that follows.
4 – N [W]	Bitmap used to enable or disable routing of sysex messages on a per-port basis. Set the appropriate bit to allow forwarding sysex messages to specific MIDI ports. Clear the appropriate bit to disallow forwarding sysex messages to specific MIDI ports. This bitmap is identical in format to that used for MIDI port routing (see RetMIDIPortRoute command). Bit 0 is port #1, bit 1 is port #2, etc. Least significant byte is first, most significant byte is last. The most significant 4 bits of each byte must be 0 which allows 4 ports to be specified per byte. Unused bits should be set to 0. The number of ports is given in the RetMIDIInfo command. The number of bytes (in the bitmap) is an even value so that the bitmap (when unpacked) is a multiple of 8 bits: number of bytes (using INTEGER math) = (((number of MIDI ports - 1) / 8) + 1) x 2

Example (device has 20 MIDI ports and supports chain mode):

0xF0, 0x00, 0x01, 0x73, 0x7E // header

```
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x17                      // command flags and ID
0x00, 0x11                      // data length (17)
0x01                            // command version number (1)
0x02                            // number of device mode blocks that follow (2)
0x06                            // block #1: size of this block (6)
0x01                            // block #1: device mode block type: sysex mode (1)
0x01                            // block #1: current sysex mode: network mode (1)
0x02                            // block #1: number of sysex modes that follow (2)
0x01                            // block #1: sysex mode #1: (network mode) (1)
0x02                            // block #1: sysex mode #2: (chain mode) (2)
0x09                            // block #2: size of this block (9)
0x02                            // block #2: device mode block type: chain route map (2)
0x06                            // block #2: bitmask length (6 bytes)
0x06                            // block #2: bitmask for MIDI ports 4 through 1
0x04                            // block #2: bitmask for MIDI ports 8 through 5
0x0C                            // block #2: bitmask for MIDI ports 12 through 9
0x03                            // block #2: bitmask for MIDI ports 16 through 13
0x08                            // block #2: bitmask for MIDI ports 20 through 17
0x00                            // block #2: bitmask for MIDI ports 24 through 21 (padding)
0xxx                            // checksum
0xF7                            // footer
```

2.24. GetUserData (Command ID = 0x18)

This command is used to retrieve user (or application) specific data that is stored in the device. The device does not use this data; consider this to be a scratchpad area for applications to store whatever data they wish in non-volatile memory. Devices that support this command will respond with a RetUserData message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x18.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x18                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

2.25. RetUserData / SetUserData (Command ID = 0x19)

RetUserData is sent by devices in response to a GetUserData command.

SetUserData is sent by a host to a device to set the user (or application) data in the device. User data is a simple array of bytes each with the most significant bit clear.

Command Data

Command flags are Answer for RetUserData, Write for SetUserData.
Command ID is 0x19.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2: index into user data array (1 – N). For RetUserData command, this is always 1 (i.e. the entire user data array is returned). For SetUserData this value can be used to offset into the user data array to write a portion of the array.

Byte #3: number of bytes that follow (1 – N). For RetUserData command, this is always the maximum number of bytes that can be stored in the device (i.e. the entire user data array is returned). For SetUserData, this can be used to write just a portion of the array.

Bytes #4-N: user data. Note that each byte must have the most-significant bit clear to be compliant with MIDI sysex messages (0x00 – 0x7F).

Example: RetUserData

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x19                      // command flags and ID
0x00, 0x07                      // data length (7)
0x01                            // command version number (1)
0x01                            // index into user data array (always 1 for RetUserData)
0x04                            // number of bytes that follow (always the maximum for RetUserData)
0x00                            // user byte at index #1
0x7F                            // user byte at index #2
0x55                            // user byte at index #3
0x02                            // user byte at index #4
0xxx                            // checksum
0xF7                            // footer
```

Example: SetUserData writing only 2 bytes in the array beginning at byte #3

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
```

```
0x40, 0x19                  // command flags and ID
0x00, 0x05                  // data length (5)
0x01                        // command version number (1)
0x03                        // index into user data array (3)
0x02                        // number of bytes that follow (2)
0x33                        // user byte at index #3
0x44                        // user byte at index #4
0xxx                        // checksum
0xF7                        // footer
```

3. MIDI Commands

The following commands are defined for protocol version = 1.

3.1. GetMIDIInfo (Command ID = 0x20)

This command is used to query a device about MIDI parameters. Devices that support this command will respond with a RetMIDIInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x20.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E  // header
0x00, 0x05                    // product ID
0x01, 0x02, 0x03, 0x04, 0x05  // serial number
0x00, 0x00                    // transaction ID
0x40, 0x20                    // command flags and ID
0x00, 0x00                    // data length
0xxx                          // checksum
0xF7                          // footer
```

3.2. RetMIDIInfo / SetMIDIInfo (Command ID = 0x21)

RetMIDIInfo is sent by devices in response to a GetMIDIInfo command. It contains basic information that a host will need to use to communicate with this device for all other MIDI related messages. A host should cache this information and use it for all further communication with this device.

SetMIDIInfo is sent by a host to a device to set the values of some of the parameters. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMIDIInfo, Write for SetMIDIInfo.
Command ID is 0x21.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Data length = 15.

Bytes #2-3: number of MIDI ports supported by this device (0 - N).
Bytes #4-5: MIDI port number that the host is using to communicate to this device (0 - N).
Byte# 6: number of DIN jack pairs (in/out) supported by this device (0 - N).
Byte# 7: number of USB device jacks supported by this device (0 - N).
Byte# 8: number of USB host jacks supported by this device (0 - N).
Byte# 9: number of ethernet jacks supported by this device (0 - N).
Byte# 10: number of USB-MIDI ports supported by each USB device jack (0 - 16).
Byte# 11: number of USB-MIDI ports supported by each USB host jack (0 - N).
Byte# 12: number of RTP-MIDI sessions supported by each ethernet jack (0 - N).
Byte# 13: number of RTP-MIDI connections supported by each RTP-MIDI session (0 - N).

Byte #14 [W]: global MIDI flags:

Bit	Description
7 - 2	reserved (always zero)
1	set if routing should be enabled between ports on multi-port USB devices (only applicable if device has USB host jacks)
0	set if running status should be enabled on DIN outputs (only applicable if device has DIN jacks)

Byte #15 [W]: maximum number of ports to use on multi-port USB devices connected to a USB host jack (only applicable if device has USB host jacks). Minimum value is 1, maximum value is the number of USB-MIDI ports supported by each USB host jack (the value in byte #11).

For command version number = 2:

Data length = 16.

Bytes #2-3: number of MIDI ports supported by this device (0 - N).
Bytes #4-5: MIDI port number that the host is using to communicate to this device (0 - N).
Byte# 6: number of DIN jack pairs (in/out) supported by this device (0 - N).
Byte# 7: number of USB device jacks supported by this device (0 - N).
Byte# 8: number of USB host jacks supported by this device (0 - N).
Byte# 9: number of ethernet jacks supported by this device (0 - N).
Byte# 10: number of USB-MIDI ports supported by each USB device jack (0 - 16).
Byte# 11: number of USB-MIDI ports supported by each USB host jack (0 - N).

Byte# 12: number of RTP-MIDI sessions supported by each ethernet jack (0 - N).
Byte# 13: number of RTP-MIDI connections supported by each RTP-MIDI session (0 - N).
Byte# 14: number of control ports supported by this device (0 - N).

Byte #15 [W]: global MIDI flags:

Bit	Description
7 - 2	reserved (always zero)
1	set if routing should be enabled between ports on multi-port USB devices (only applicable if device has USB host jacks)
0	set if running status should be enabled on DIN outputs (only applicable if device has DIN jacks)

Byte #16 [W]: maximum number of ports to use on multi-port USB devices connected to a USB host jack (only applicable if device has USB host jacks). Minimum value is 1, maximum value is the number of USB-MIDI ports supported by each USB host jack (the value in byte #11).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x00, 0x21                     // command flags and ID
0x00, 0x10                     // data length (16)
0x02                           // command version number (2)
0x00, 0x14                     // number of MIDI ports (20)
0x00, 0x01                     // host is communicating on MIDI port #1
0x02                           // device has 2 sets of DIN jacks
0x02                           // device has 2 USB device jacks
0x01                           // device has 1 USB host jack
0x01                           // device has 1 ethernet jack
0x04                           // device supports 4 MIDI ports on each USB device jack
0x08                           // device supports 8 MIDI ports on each USB host jack
0x04                           // device supports 4 RTP-MIDI sessions on each ethernet jack
0x01                           // device supports 1 RTP-MIDI connection on each RTP-MIDI session
0x01                           // device supports 1 control port
0x01                           // routing between multi-port is disabled, running status is enabled
0x04                           // use 4 ports maximum on multi-port USB devices
0xxx                           // checksum
0xF7                           // footer
```

3.3. GetMIDIPortInfo (Command ID = 0x22)

This command is used to query a device about a specific MIDI port. Devices that support this command will respond with a RetMIDIPortInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x22.
Data length is 2.

Bytes #1-2: MIDI port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x22                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // port ID
0xxx                            // checksum
0xF7                            // footer
```

3.4. RetMIDIPortInfo / SetMIDIPortInfo (Command ID = 0x23)

RetMIDIPortInfo is sent by devices in response to a GetMIDIPortInfo command.

SetMIDIPortInfo is sent by a host to a device to set the values of some of the port parameters. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMIDIPortInfo, Write for SetMIDIPortInfo.
Command ID is 0x23.
Data length depends on command version number. For version = 1, minimum length is 10, actual length depends on the port name (a 7-bit ASCII string, not NULL terminated). Use the data length field to determine the string length.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1 or 2:

Bytes #2-3: port ID (1 - N).

Byte #4: port type:

Value	Description
1	DIN
2	USB device
3	USB host
4	Ethernet
5	Control

Bytes #5-8: port info, depends on port type (byte #4):

Type	Description
DIN	Byte #5: DIN jack # (1 - N) Bytes #6-8: reserved (always 0)
USB device	Byte #5: USB device jack # (1 - N) Byte #6: device port # (1 - 16) Bytes #7-8: reserved (always 0)
USB host	Byte #5: USB host jack # (1 - N) Byte #6: USB host jack port # (1 - N) Byte #7-8: reserved (always 0)
Ethernet	Byte #5: ethernet jack # (1 - N) Byte #6: session # (1 - N) Byte #7-8: reserved (always 0)
Control	Byte #5: control port # (1 - N) Byte #6: control port type: 1 = Automation Control Byte #7-8: reserved (always 0)

Byte #9: maximum length allowed for port name, 0 if read-only (0 - 127).

Byte #10: port flags:

Bit	Description
7	reserved (always zero)

Bit	Description
6	set if this port can be used for firmware updates [only for version #2, always 0 for earlier versions]
5 - 4	reserved (always zero)
3	set if port has output [only for command version #2, always 0 for earlier versions because all ports were assumed to have an output]
2	set if port has input [only for command version #2, always 0 for earlier versions because all ports were assumed to have an input]
1 [W]	set if port output is enabled
0 [W]	set if port input is enabled

Bytes #11-N [W]: port name, 7-bit ASCII string, not NULL terminated. Name must follow the same rules as used for device name (see RetInfo command).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x23                      // command flags and ID
0x00, 0x0E                      // data length (14)
0x02                            // command version number (2)
0x00, 0x01                      // port ID
0x01                            // port type = DIN
0x01                            // port is first set of DIN jacks
0x00, 0x00, 0x00                // reserved for DIN port
0x0F                            // maximum length allowed for port name (15 ASCII characters)
0x0F                            // port flags: input and output both exist and are both enabled
0x44, 0x49, 0x4E, 0x31          // port name, the ASCII string “DIN1”
0xxx                            // checksum
0xF7                            // footer
```

3.5. GetMIDIPortFilter (Command ID = 0x24)

This command is used to query a device about a specific MIDI port’s filters. Devices that support this command will respond with a RetMIDIPortFilter message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x24.

Data length is 3.

Bytes #1-2: MIDI port ID (1 - N).
Byte #3: filter ID:

Filter ID	Description
1	input filter
2	output filter

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x24                      // command flags and ID
0x00, 0x03                      // data length
0x00, 0x01                      // port ID
0x01                            // filter ID (input)
0xxx                            // checksum
0xF7                            // footer
```

3.6. RetMIDIPortFilter / SetMIDIPortFilter (Command ID = 0x25)

RetMIDIPortFilter is sent by devices in response to a GetMIDIPortFilter command.

SetMIDIPortFilter is sent by a host to a device to set the values of some of the port's filter parameters. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMIDIPortFilter, Write for SetMIDIPortFilter.
Command ID is 0x25.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: port ID (1 - N).
Byte #4: filter ID (1 - N), (input = 1, output = 2).
Byte #5: maximum number of controller filters supported by this filter (0 - 127).

Bytes #6-7 [W]: bitmaps indicating filter status for the following system messages:

Byte	Bit	Description
6	7 – 1	always zero
	0	set if port should filter MIDI reset events (0xFF)
7	7	always zero
	6	set if port should filter MIDI active sensing events (0xFE)
	5	set if port should filter MIDI realtime events (0xF8-0xFD)
	4	set if port should filter MIDI tune request events (0xF6)
	3	set if port should filter MIDI song select events (0xF3)
	2	set if port should filter MIDI song position pointer events (0xF2)
	1	set if port should filter MIDI time code (MTC) events (0xF1)
	0	set if port should filter MIDI system exclusive events (0xF0, 0xF7)

Bytes #8-23 [W]: bitmap indicating filter status for channel messages. One byte for each of the 16 MIDI channels (MIDI channel #1 is the first byte, MIDI channel #16 is the last byte). Note that setting the filter for controller changes will filter out all controllers on that MIDI channel; to filter out specific controllers use the controller filters section (bytes 24 and up). Each bitmap for bytes #8-23 is as follows:

Bit	Description
7 - 6	always zero
5	set if port should filter MIDI pitch bend events (0xEn)
4	set if port should filter MIDI channel pressure events [mono aftertouch] (0xDn)
3	set if port should filter MIDI program change events (0xCn)
2	set if port should filter MIDI control change events (0xBn)
1	set if port should filter MIDI poly key pressure events [poly aftertouch] (0xAn)
0	set if port should filter MIDI note on/off events (0x8n, 0x9n)

Bytes #24-N [W]: controller filters. Bytes 24 and up are used only if byte #5 is not zero. Five bytes are used for each controller filter as follows:

Byte	Description
1 - 4	MIDI channel bitmap, specific to this controller filter
5	controller ID (0 - 127)

The MIDI channel bitmap has the following format:

Byte	Bit	Description
1	7 - 4	always zero
	3	set if port should filter MIDI messages on channel 16
	2	set if port should filter MIDI messages on channel 15
	1	set if port should filter MIDI messages on channel 14
	0	set if port should filter MIDI messages on channel 13
2	7 - 4	
	3	set if port should filter MIDI messages on channel 12
	2	set if port should filter MIDI messages on channel 11
	1	set if port should filter MIDI messages on channel 10
	0	set if port should filter MIDI messages on channel 9
3	7 - 4	always zero
	3	set if port should filter MIDI messages on channel 8
	2	set if port should filter MIDI messages on channel 7
	1	set if port should filter MIDI messages on channel 6
	0	set if port should filter MIDI messages on channel 5
4	7 - 4	always zero
	3	set if port should filter MIDI messages on channel 4
	2	set if port should filter MIDI messages on channel 3
	1	set if port should filter MIDI messages on channel 2

Byte	Bit	Description
	0	set if port should filter MIDI messages on channel 1

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x00, 0x25                     // command flags and ID
0x00, 0x21                     // data length (33)
0x01                           // command version number (1)
0x00, 0x01                     // port ID
0x01                           // filter ID (input)
0x02                           // number of controller filters (2)
0x00, 0x40                     // filter active sensing
0x20                           // filter pitch bend on MIDI channel 1
0x10                           // filter mono aftertouch on MIDI channel 2
0x04                           // filter all controllers on MIDI channel 3
0x3F                           // filter all channel messages on MIDI channel 4
0x20                           // filter pitch bend on MIDI channel 5
0x10                           // filter mono aftertouch on MIDI channel 6
0x04                           // filter all controllers on MIDI channel 7
0x3F                           // filter all channel messages on MIDI channel 8
0x20                           // filter pitch bend on MIDI channel 9
0x10                           // filter mono aftertouch on MIDI channel 10
0x04                           // filter all controllers on MIDI channel 11
0x3F                           // filter all channel messages on MIDI channel 12
0x20                           // filter pitch bend on MIDI channel 13
0x10                           // filter mono aftertouch on MIDI channel 14
0x04                           // filter all controllers on MIDI channel 15
0x3F                           // filter all channel messages on MIDI channel 16
0x00, 0x00, 0x00, 0x01, 0x07   // filter controller #7 (volume on channel 1)
0x00, 0x00, 0x00, 0x00, 0x40   // filter controller #64 (sustain pedal, filter disabled)
0xxx                           // checksum
0xF7                           // footer
```

3.7. GetMIDIPortRemap (Command ID = 0x26)

This command is used to query a device about a specific MIDI port's remap. Devices that support this command will respond with a RetMIDIPortRemap message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x26.

Data length is 3.

Bytes #1-2: MIDI port ID (1 - N).
Byte #3: remap ID:

Remap ID	Description
1	input remap
2	output remap

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x26                      // command flags and ID
0x00, 0x03                      // data length
0x00, 0x01                      // port ID
0x01                            // remap ID (input)
0xxx                            // checksum
0xF7                            // footer
```

3.8. RetMIDIPortRemap / SetMIDIPortRemap (Command ID = 0x27)

RetMIDIPortRemap is sent by devices in response to a GetMIDIPortRemap command.

SetMIDIPortRemap is sent by a host to a device to set the values of some of the port's remap parameters. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMIDIPortRemap, Write for SetMIDIPortRemap.
Command ID is 0x27.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: port ID (1 - N).
Byte #4: remap ID (input = 1, output = 2).
Byte #5: maximum number of controller remaps supported by this remap (0 - 127).

Bytes #6-37 [W]: bitmap indicating remap status and remap channel for channel messages. Two bytes for each of the 16 MIDI channels (MIDI channel #1 is the first two bytes, MIDI channel #16 is the last two bytes). Each bitmap for bytes #6-37 is as follows:

Byte	Bit	Description
1	7 - 6	always zero
	5	set if port should remap MIDI pitch bend events (0xEn)
	4	set if port should remap MIDI channel pressure events [mono aftertouch] (0xDn)
	3	set if port should remap MIDI program change events (0xCn)
	2	set if port should remap MIDI control change events (0xBn)
	1	set if port should remap MIDI poly key pressure events [poly aftertouch] (0xAn)
	0	set if port should remap MIDI note on/off events (0x8n, 0x9n)
2	7 - 4	reserved (always zero)
	3 - 0	MIDI channel number for remap. MIDI channels are defined as 1 through 16 but the value used is 0 through 15 (i.e. MIDI channel 1 has value 0, MIDI channel 2 has value 1, etc.)

Bytes #38-N [W]: controller remap flags and value pairs. Bytes 38 and up are only used if byte #5 in the header is not zero. 6 bytes are used for each controller remap as follows:

Byte	Description
1 - 4	MIDI channel bitmap, specific to this controller remap
5	controller source ID (0 - 127)
6	controller destination ID (0 - 127)

The MIDI channel bitmap has the following format:

Byte	Bit	Description
1	7 - 4	always zero
	3	set if port should remap MIDI messages on channel 16

Byte	Bit	Description
	2	set if port should remap MIDI messages on channel 15
	1	set if port should remap MIDI messages on channel 14
	0	set if port should remap MIDI messages on channel 13
2	7 - 4	
	3	set if port should remap MIDI messages on channel 12
	2	set if port should remap MIDI messages on channel 11
	1	set if port should remap MIDI messages on channel 10
	0	set if port should remap MIDI messages on channel 9
3	7 - 4	always zero
	3	set if port should remap MIDI messages on channel 8
	2	set if port should remap MIDI messages on channel 7
	1	set if port should remap MIDI messages on channel 6
	0	set if port should remap MIDI messages on channel 5
4	7 - 4	always zero
	3	set if port should remap MIDI messages on channel 4
	2	set if port should remap MIDI messages on channel 3
	1	set if port should remap MIDI messages on channel 2
	0	set if port should remap MIDI messages on channel 1

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x27                      // command flags and ID
0x00, 0x31                      // data length (49)
0x01                            // command version number (1)
0x00, 0x01                      // port ID
0x01                            // remap ID (input)
0x02                            // number of controller remaps (2)
```

```
0x3F, 0x09                  // channel 1: remap all messages to channel 10
0x00, 0x01                  // channel 2: don't remap any messages
0x01, 0x0F                  // channel 3: remap note events to channel 16
0x20, 0x00                  // channel 4: remap pitch bend to channel 1
0x00, 0x04                  // channel 5: don't remap any messages
0x3F, 0x01                  // channel 6: remap all messages to channel 2
0x01, 0x02                  // channel 7: remap mono aftertouch to channel 3
0x00, 0x07                  // channel 8: don't remap any messages
0x3F, 0x09                  // channel 9: remap all messages to channel 10
0x00, 0x09                  // channel 10: don't remap any messages
0x01, 0x0F                  // channel 11: remap note events to channel 16
0x20, 0x00                  // channel 12: remap pitch bend to channel 1
0x00, 0x0C                  // channel 13: don't remap any messages
0x3F, 0x01                  // channel 14: remap all messages to channel 2
0x01, 0x02                  // channel 15: remap mono aftertouch to channel 3
0x00, 0x0F                  // channel 16: don't remap any messages
0x0F, 0x0F, 0x0F, 0x0F,     // remap controller #1 to controller #2 (all channels)
0x01, 0x02
0x00, 0x00, 0x00, 0x03,     // remap controller #3 to controller #4 (channels 1 & 2)
0x03, 0x04
0xxx                        // checksum
0xF7                        // footer
```

3.9. GetMIDIPortRoute (Command ID = 0x28)

This command is used to query a device about a specific MIDI port's routing. Devices that support this command will respond with a RetMIDIPortRoute message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x28.
Data length is 2.

Bytes #1-2: MIDI port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x28                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // port ID
0xxx                            // checksum
0xF7                            // footer
```

3.10. RetMIDIPortRoute / SetMIDIPortRoute (Command ID = 0x29)

RetMIDIPortRoute is sent by devices in response to a GetMIDIPortRoute command.

SetMIDIPortRoute is sent by a host to a device to set the port's routing. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMIDIPortRoute, Write for SetMIDIPortRoute.
Command ID is 0x29.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: port ID (1 - N).
Bytes #4-N [W]: bitmap indicating port routing of this port to all other ports. Bit 0 is port #1, bit 1 is port #2, etc. Least significant byte is first, most significant byte is last. The most significant 4 bits of each byte must be 0 which allows 4 ports to be specified per byte. Unused bits should be set to 0. The number of ports is given in the RetMIDIInfo command. The number of bytes (in the bitmap) is an even value so that the bitmap (when unpacked) is a multiple of 8 bits:

number of bytes (using INTEGER math) = (((number of ports - 1) / 8) + 1) x 2

Example: device has 20 ports, route port #1 to ports 2, 3, 7, 11-14, 19, 20

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x29                      // command flags and ID
0x00, 0x09                      // data length (9)
0x01                            // command version number (1)
0x00, 0x01                      // port ID
0x06                            // routing for ports 4 through 1
0x04                            // routing for ports 8 through 5
0x0C                            // routing for ports 12 through 9
0x03                            // routing for ports 16 through 13
0x08                            // routing for ports 20 through 17
0x00                            // routing for ports 24 through 21 (padding)
0xxx                            // checksum
0xF7                            // footer
```

3.11. GetMIDIPortDetail (Command ID = 0x2A)

This command is used to query a device about details related to a specific MIDI port. Devices that support this command will respond with a RetMIDIPortDetail message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x2A.
Data length is 2.

Bytes #1-2: MIDI port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x2A                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // port ID
0xxx                            // checksum
0xF7                            // footer
```

3.12. RetMIDIPortDetail / SetMIDIPortDetail (Command ID = 0x2B)

RetMIDIPortDetail is sent by devices in response to a GetMIDIPortDetail command.

SetMIDIPortDetail is sent by a host to a device to set the MIDI port details of USB host ports. DIN ports, USB device ports, and ethernet ports cannot be modified using this command. Writeable parameters for USB host ports are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored. The host only needs to send up to (and including) byte #7, all bytes beyond #7 are read-only and are ignored.

Command Data

Command flags are Answer for RetMIDIPortDetail, Write for SetMIDIPortDetail.
Command ID is 0x2B.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: MIDI port ID (1 - N).

Byte #4: port type:

Value	Description
1	DIN
2	USB device

Value	Description
3	USB host
4	Ethernet
5	Control

Bytes #5-N: port detail, depends on port type (byte #4):

Type	Description
DIN	No details are available for DIN ports.
USB device	Byte #5: host type: 0 = no host, 1 = Mac/PC, 2 = iOS device Byte #6: length of host name field that follows (0 - N) Bytes #7-N: host name, 7-bit ASCII string, not NULL terminated Host name is only applicable for host type = iOS device. For Mac/PC hosts (or no host at all), length field is always 0x00 and host name is empty.
USB host	Byte #5 [W]: Flags: bits 7-1 = always zero bit 0 = set if port is reserved for a specific vendor ID and product ID Byte #6 [W]: USB host ID (1 - N), is 0 if no device is being hosted on this port or if a port is inactive because a reserved device could not be connected. Byte #7 [W: Hosted device's MIDI port number (1 - 16), is 0 if no device is hosted or reserved on this port. Byte #8-10: Hosted or reserved device's USB vendor ID (16 bit value encoded in 3 bytes), is 0 if no device is hosted or reserved on this port. Byte #11-13: Hosted or reserved device's USB product ID (16 bit value encoded in 3 bytes, is 0 if no device is hosted or reserved on this port). Byte #14: Length of vendor name field that follows (0 - N). Bytes #X: Vendor name, 7-bit ASCII string, not NULL terminated (from USB descriptor of hosted device). Byte #X: Length of product name field that follows (0 - N). Bytes #X: Product name, 7-bit ASCII string, not NULL terminated (from USB descriptor of hosted device). The maximum value for USB host ID (byte #6) is the same as the number of USB-MIDI ports supported by each USB host jack (byte #11 in RetMIDIInfo command). The maximum value for Device MIDI port number (byte #7) depends on the number of MIDI ports supported by the hosted device (bytes #4-5 in RetUSBHostMIDIDeviceDetail command).

Type	Description
Ethernet	Bytes #5-7: RTP-MIDI port number (16 bit value encoded in 3 bytes) Byte #8: number of active RTP-MIDI connections on this MIDI port (RTP-MIDI session) Byte #9: Length of session name (Bonjour name) field that follows (0 - N). Bytes #10-N: Session name (Bonjour name), 7-bit ASCII string, not NULL terminated. Session name is only valid if the ethernet port is connected to a network, is active, and has finished Bonjour name resolution.
Control	No details are available for control ports.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x2B                      // command flags and ID
0x00, 0x0A                      // data length (10)
0x01                            // command version number (1)
0x00, 0x03                      // port ID
0x02                            // port type = USB device
0x02                            // host type = iOS device
0x04                            // length of ASCII string that follows (4)
0x69, 0x50, 0x61, 0x64          // host name, the ASCII string “iPad”
0xxx                            // checksum
0xF7                            // footer
```

3.13. GetRTPMIDIConnectionDetail (Command ID = 0x2C)

This command is used to query a device about details related to a specific RTP-MIDI connection. Devices that support this command will respond with a RetRTPMIDIConnectionDetail message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x2C.
Data length is 3.

Bytes #1-2: MIDI port ID (1 - N).
Byte #3: RTP-MIDI connection number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
```

```
0x00, 0x00                  // transaction ID
0x40, 0x2C                  // command flags and ID
0x00, 0x03                  // data length
0x00, 0x3D                  // port ID (61)
0x01                        // connection number (1)
0xxx                        // checksum
0xF7                        // footer
```

3.14. RetRTPMIDIConnectionDetail (Command ID = 0x2D)

RetRTPMIDIConnectionDetail is sent by devices in response to a GetRTPMIDIConnectionDetail command.

Command Data

Command flags are Answer.
Command ID is 0x2D.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: MIDI port ID (1 - N).
Byte #4: RTP-MIDI connection number (1 - N).
Bytes #5-9: IP address of the device on the other side of this connection. A four-byte big-endian value encoded in five-bytes (similar to device ID) so that the most significant bit of each byte is 0.
Bytes #10-12: RTP-MIDI port number of the device on the other side of this connection (16 bit value encoded in 3 bytes).
Byte #13: Length of session name (Bonjour name) field that follows (0 - N).
Bytes #14-N: Session name (Bonjour name) of the device on the other side of this connection, 7-bit ASCII string, not NULL terminated.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x2D                      // command flags and ID
0x00, 0x11                      // data length (17)
0x01                            // command version number (1)
0x00, 0x3D                      // port ID (61)
0x01                            // connection number (1)
0x0C, 0x05, 0x20, 0x02, 0x65    // IP address (192.168.1.101)
0x00, 0x27, 0x0C                // port number (5004)
0x04                            // length of session name string that follows (4)
0x4D, 0x49, 0x44, 0x49          // the ASCII string “MIDI”
0xxx                            // checksum
0xF7                            // footer
```

3.15. GetUSBHostMIDIDeviceDetail (Command ID = 0x2E)

This command is used to query a device about details related to USB MIDI devices that are being hosted by USB host jacks. Devices that support this command will respond with a RetUSBHostMIDIDeviceDetail message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x2E.
Data length is 2.

Byte #1: USB host jack # (1 - N). Maximum value is the number of USB host jacks supported by this device (byte #8 in RetMIDIInfo command).
Byte #2: USB host ID (1 - N). Maximum value is the number of USB-MIDI ports supported by each USB host jack (byte #11 in RetMIDIInfo command).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x40, 0x2E                     // command flags and ID
0x00, 0x02                     // data length
0x01                           // USB host jack # (1)
0x01                           // USB host ID (1)
0xxx                           // checksum
0xF7                           // footer
```

3.16. RetUSBHostMIDIDeviceDetail (Command ID = 0x2F)

RetUSBHostMIDIDeviceDetail is sent by devices in response to a GetUSBHostMIDIDeviceDetail command.

Command Data

Command flags are Answer.
Command ID is 0x2F.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2: USB host jack # (1 - N). Maximum value is the number of USB host jacks supported by this device (byte #8 in RetMIDIInfo command).
Byte #3: USB host ID (1 - N). Maximum value is the number of USB-MIDI ports supported by each USB host jack (byte #11 in RetMIDIInfo command).
Byte #4: Number of MIDI IN ports supported by the hosted device (1 - 16), is 0 if no device is connected.

Byte #5: Number of MIDI OUT ports supported by the hosted device (1 - 16), is 0 if no device is connected.
Byte #6-8: Hosted device's USB vendor ID (16 bit value encoded in 3 bytes), is 0 if no device is connected.
Byte #9-11: Hosted device's USB product ID (16 bit value encoded in 3 bytes), is 0 if no device is connected.
Byte #12: Length of vendor name field that follows (0 - N).
Bytes #X: Vendor name, 7-bit ASCII string, not NULL terminated (from USB descriptor of hosted device).
Byte #X: Length of product name field that follows (0 - N).
Bytes #X: Product name, 7-bit ASCII string, not NULL terminated (from USB descriptor of hosted device).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x2F                      // command flags and ID
0x00, 0x15                      // data length (21)
0x01                            // command version number (1)
0x01                            // USB host jack # (1)
0x02                            // USB host ID (2)
0x04                            // number of MIDI IN ports (4)
0x04                            // number of MIDI OUT ports (4)
0x00, 0x46, 0x21                // vendor ID (0x2321)
0x00, 0x00, 0x0F                // product ID (0x000F)
0x04                            // length of vendor name string that follows (4)
0x49, 0x43, 0x4F, 0x4E          // the ASCII string “ICON”
0x04                            // length of product name string that follows (4)
0x49, 0x43, 0x4D, 0x32          // the ASCII string “ICM2”
0xxx                            // checksum
0xF7                            // footer
```

3.17. GetMIDIMonitor (Command ID = 0x70)

This command is used to query a device for the most recent MIDI monitor values (MIDI activity indicators) for all MIDI ports on the device. Devices that support this command will respond with a RetMIDIMonitor message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x70.
Data length is 1.

Byte #1: bitmap indicating which MIDI monitor values should be returned from the device:

Bit	Description
7 - 2	always zero
1	MIDI outputs
0	MIDI inputs

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x70                      // command flags and ID
0x00, 0x01                      // data length
0x03                            // get MIDI monitor values for inputs and outputs
0xxx                            // checksum
0xF7                            // footer
```

3.18. RetMIDIMonitor (Command ID = 0x71)

RetMIDIMonitor is sent by devices in response to a GetMIDIMonitor command.

Command Data

Command flags are Answer.
Command ID is 0x71.
Data length depends on the number of MIDI ports.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2: number of MIDI monitor blocks that follow.
Bytes #3-N: MIDI monitor blocks:

Byte	Description
1	Bitmap indicating which monitor values are in this block: bits 7 - 2: always zero bit 1: MIDI outputs bit 0: MIDI inputs
2 - N	Bitmap indicating MIDI activity on all ports. Bit 0 is port #1, bit 1 is port #2, etc.

Byte	Description
	Least significant byte is first, most significant byte is last. The most significant 4 bits of each byte must be 0 which allows 4 ports to be specified per byte. Unused bits should be set to 0. The number of ports is given in the RetMIDIInfo command. The number of bytes (in the bitmap) is an even value so that the bitmap (when unpacked) is a multiple of 8 bits: number of bytes (using INTEGER math) = (((number of ports - 1) / 8) + 1) x 2

Example (device has 20 ports):

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x71                      // command flags and ID
0x00, 0x10                      // data length (16)
0x01                            // command version number (1)
0x02                            // number of MIDI monitor blocks that follow (2)
0x01                            // block #1: MIDI inputs (0x01)
0x06                            // MIDI input activity for ports 4 through 1
0x04                            // MIDI input activity for ports 8 through 5
0x0C                            // MIDI input activity for ports 12 through 9
0x03                            // MIDI input activity for ports 16 through 13
0x08                            // MIDI input activity for ports 20 through 17
0x00                            // MIDI input activity for ports 24 through 21 (padding)
0x02                            // block #2: MIDI outputs (0x02)
0x07                            // MIDI output activity for ports 4 through 1
0x00                            // MIDI output activity for ports 8 through 5
0x02                            // MIDI output activity for ports 12 through 9
0x05                            // MIDI output activity for ports 16 through 13
0x0D                            // MIDI output activity for ports 20 through 17
0x00                            // MIDI output activity for ports 24 through 21 (padding)
0xxx                            // checksum
0xF7                            // footer
```

3.19. GetRTPMIDIConnectionParm (Command ID = 0x7E)

This command is used to query a device about parameters related to a specific RTP-MIDI connection. Devices that support this command will respond with a RetRTPMIDIConnectionParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x7E.
Data length is 3.

Bytes #1-2: MIDI port ID (1 - N).

Byte #3: RTP-MIDI connection number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x7E                      // command flags and ID
0x00, 0x03                      // data length
0x00, 0x3D                      // port ID (61)
0x01                            // connection number (1)
0xxx                            // checksum
0xF7                            // footer
```

3.20. RetRTPMIDIConnectionParm (Command ID = 0x7F)

RetRTPMIDIConnectionParm is sent by devices in response to a GetRTPMIDIConnectionParm command.

SetRTPMIDIConnectionParm is sent by a host to a device to set the parameters for a specific RTP-MIDI connection. The host only needs to send up to and including byte #6 (flags) if the IP address, port number, and session name are not being changed.

Command Data

Command flags are Answer for RetRTPMIDIConnectionParm, Write for SetRTPMIDIConnectionParm.
Command ID is 0x7F.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: MIDI port ID (1 - N).
Byte #4: RTP-MIDI connection number (1 - N).
Byte #5: maximum length allowed for session name (1 - N). Read-only.
Byte #6: flags:

Bit	Description
7 - 3	reserved (always zero)
2	Mode (only used if Role = initiator): 0 = use IP address and port number 1 = use session name
1	Role: 0 = responder

Bit	Description
	1 = initiator
0	Enable: 0 = disable 1 = enable

The remaining bytes only need to be sent if the IP address, port number, or session name are being modified.

Bytes #7-11: IP address of the device on the other side of this connection. A four-byte big-endian value encoded in five-bytes (similar to device ID) so that the most significant bit of each byte is 0. Only used if Role = 1 and Mode = 0.
Bytes #12-14: RTP-MIDI port number of the device on the other side of this connection (16 bit value encoded in 3 bytes). Only used if Role = 1 and Mode = 0.
Byte #15: Length of session name (Bonjour name) field that follows (0 - N).
Bytes #16-N: Session name (Bonjour name) of the device on the other side of this connection, 7-bit ASCII string, not NULL terminated. Only used if Role = 1 and Mode = 1;

Example (not changing IP address, port number or session name):

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x7F                      // command flags and ID
0x00, 0x06                      // data length (6)
0x01                            // command version number (1)
0x00, 0x3D                      // port ID (61)
0x01                            // connection number (1)
0x13                            // maximum length allowed for session name (19)
0x01                            // flags: enable, responder role
0xxx                            // checksum
0xF7                            // footer
```

Example (change everything):

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x7F                      // command flags and ID
0x00, 0x13                      // data length (19)
0x01                            // command version number (1)
0x00, 0x3D                      // port ID (61)
0x01                            // connection number (1)
0x13                            // maximum length allowed for session name (19)
0x07                            // flags: enable, initiator role, use session name
0x0C, 0x05, 0x20, 0x02, 0x65    // IP address (192.168.1.101)
```

```
0x00, 0x27, 0x0C          // port number (5004)
0x04                      // length of session name string that follows (4)
0x4D, 0x49, 0x44, 0x49    // the ASCII string “MIDI”
0xxx                      // checksum
0xF7                      // footer
```

4. Audio Commands V2

The following commands are defined for protocol version = 1.

4.1. GetAudioGlobalParm (Command ID = 0x40)

This command is used to query a device about global audio parameters. Devices that support this command will respond with a RetAudioGlobalParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x40.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x40, 0x40                     // command flags and ID
0x00, 0x00                     // data length
0xxx                           // checksum
0xF7                           // footer
```

4.2. RetAudioGlobalParm / SetAudioGlobalParm (Command ID = 0x41)

RetAudioGlobalParm is sent by devices in response to a GetAudioGlobalParm command. It contains information that a host will need to use to communicate with this device for other audio related messages. A host should cache this information and use it for further communication with this device.

SetAudioGlobalParm is sent by a host to a device to set the current audio configuration. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored. Parameters in RED underline require the device to be reset before taking effect.

Command Data

Command flags are Answer for RetAudioGlobalParm, Write for SetAudioGlobalParm.
Command ID is 0x41.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The host only needs to send up to (and including) byte #10, all bytes beyond #10 are read-only and are ignored.

Byte #2-3: total number of audio ports supported by this device (0 - N).
Byte #4: minimum number of audio frames that can be buffered (1 - N).
Byte #5: maximum number of audio frames that can be buffered (1 - N).
Byte #6 [W]: current number of audio frames to buffer (1 - N).
Byte #7: minimum allowed value for sync factor (1 - N).
Byte #8: maximum allowed value for sync factor (1 - N).
Byte #9 [W]: current sync factor value (1 - N).
Byte #10 [W]: number of the currently active audio configuration (1 - N).
Byte #11: number of configuration blocks that follow (0 - N).

Bytes #12-N: audio configurations blocks. Bytes 12 and up are used only if byte #11 is not zero. Three bytes are used for each audio configuration block as follows:

Byte	Description
1	audio configuration number (1 - N)
2	bit depth code: 1 = 4 bit, 2 = 8 bit, 3 = 12 bit, 4 = 16 bit bit depth code: 5 = 20 bit, 6 = 24 bit, 7 = 28 bit, 8 = 32 bit
3	sample rate code: 1 = 11025, 3 = 22050, 5 = 44100, 7 = 88200, 9 = 176400 sample rate code: 2 = 12000, 4 = 24000, 6 = 48000, 8 = 96000, 10 = 192000

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x41                      // command flags and ID
0x00, 0x1D                      // data length (29)
0x01                            // command version number (1)
0x00, 0x06                      // device has 6 audio ports
0x01, 0x04, 0x02                // minimum, maximum, and current # of audio frames to buffer
0x01, 0x04, 0x02                // minimum, maximum, and current sync factor
0x02                            // currently active audio configuration is #2
0x06                            // number of audio configuration blocks that follow (6)
0x01, 0x04, 0x05                // configuration block #1: 16 bit, 44100
0x02, 0x04, 0x06                // configuration block #2: 16 bit, 48000
0x03, 0x04, 0x07                // configuration block #3: 16 bit, 88200
0x04, 0x04, 0x08                // configuration block #4: 16 bit, 96000
0x05, 0x06, 0x05                // configuration block #5: 24 bit, 44100
```

```
0x06, 0x06, 0x06        // configuration block #6: 24 bit, 48000
0xxx                    // checksum
0xF7                    // footer
```

4.3. GetAudioPortParm (Command ID = 0x42)

This command is used to query a device about a specific audio port. Devices that support this command will respond with a RetAudioPortParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x42.
Data length is 2.

Bytes #1-2: audio port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x42                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // audio port ID
0xxx                            // checksum
0xF7                            // footer
```

4.4. RetAudioPortParm / SetAudioPortParm (Command ID = 0x43)

RetAudioPortParm is sent by devices in response to a GetAudioPortParm command.

SetAudioPortParm is sent by a host to a device to set the values of some of the port parameters. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored. Parameters in RED underline require the device to be reset before taking effect.

Note that audio ports can have both inputs (sources) and outputs (destinations). These are always referenced with respect to the device, not the host. For example, if a device has 4 microphone inputs, 4 line outs and 1 headphone jack this would be reported as a single analogue audio port with 4 sources (the 4 microphone inputs) and 6 destinations (the 4 line outs plus 2 channels for headphone left/right).

For USB host ports, inputs to a hosted audio device are outputs (destinations) from the iConnectivity device and outputs from the hosted audio device are inputs (sources) to the iConnectivity device. For example, if a hosted audio device has 2 microphone inputs and 4 line outs this would be reported on the USB host port as having 2 sources (the 2 microphone inputs) and 4 destinations (the 4 line outs).

USB device ports are a bit trickier because the number of audio channels on that USB port can be defined by the user and the host swaps the definition of inputs and outputs. For example, if a USB port is defined

to have 4 inputs (sources) and 6 outputs (destinations) this means there are 4 channels of audio data coming into the device on that USB port and 6 channels of audio data coming out of the device on that USB port. When viewed from the host's perspective (i.e. a Mac or PC) this will appear to be the opposite: 6 inputs (to the computer) and 4 outputs (from the computer).

Command Data

Command flags are Answer for RetAudioPortParm, Write for SetAudioPortParm.
Command ID is 0x43.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The port configuration blocks are read-only, thus the host can skip sending these by setting byte #7 to 0x00.

Bytes #2-3: audio port ID (1 - N).
Byte #4: port type:

Value	Description
2	USB device
3	USB host
4	ethernet
5	analogue

Byte #5 [W]: number of destination (output) channels to use for this port (0 - N), must be a legal value for the current audio configuration.
Byte #6 [W]: number of source (input) channels to use for this port (0 - N), must be a legal value for the current audio configuration.
Byte #7: number of port configuration blocks that follow (0 - N).

Bytes #8-N: port configurations blocks, only included if byte #7 is not zero. There is one port configuration block for each audio configuration (from RetAudioGlobalParm command). Six bytes are used for each port configuration block as follows:

Byte	Description
1	audio configuration number (1 - N), from RetAudioGlobalParm command

Byte	Description
2	maximum number of audio channels (0 - N) allowed for this port (for this audio configuration number), the number of source (input) and destination (output) channels combined must not exceed this value
3	minimum number of destination (output) channels allowed for this port (0 - N)
4	maximum number of destination (output) channels allowed for this port (0 - N)
5	minimum number of source (input) channels allowed for this port (0 - N)
6	maximum number of source (input) channels allowed for this port (0 - N)

Byte #N: maximum length allowed for port name, 0 if read-only (0 - 127).
Byte #N [W]: length of port name field that follows (0 - N).
Bytes #N-N [W]: port name, 7-bit ASCII string, not NULL terminated. Name must follow the same rules as used for device name (see RetInfo command).

Bytes #N-N: port info, depends on port type (byte #4):

Type	Description
USB device	Byte #1: USB device jack # (1 - N) Byte #2: Flags: bits 7 - 4: reserved (always 0) bit 3 [W]: set if port is currently enabled for audio with iOS device bit 2 [W]: set if port is currently enabled for audio with PC/Mac bit 1: set if port supports audio with iOS devices bit 0: set if port supports audio with PC/Mac
USB host	Byte #1: USB host jack # (1 - N) Byte #2: device number on this USB host jack (1 - N), USB host jacks may support several audio devices
ethernet	Byte #1: ethernet jack # (1 - N) Byte #2: device number on this ethernet jack (1 - N), ethernet jacks may support several audio devices
analogue	Byte #1: analogue port # (1 - N)

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E        // header
0x00, 0x05                          // product ID
0x01, 0x02, 0x03, 0x04, 0x05        // serial number
```



```
0x00, 0x00                          // transaction ID
0x00, 0x43                          // command flags and ID
0x00, 0x2D                          // data length (45)
0x01                                // command version number (1)
0x00, 0x01                          // audio port ID
0x02                                // port type = USB device
0x04                                // number of destination (output) channels to use for this port (4)
0x04                                // number of source (input) channels to use for this port (4)
0x06                                // number of port configuration blocks that follow (6)
0x01, 0x08, 0x01, 0x07, 0x01, 0x07  // port configuration block #1 (16/44): 8 max, 1/7 out, 1/7 in
0x02, 0x08, 0x01, 0x07, 0x01, 0x07  // port configuration block #2 (16/48): 8 max, 1/7 out, 1/7 in
0x03, 0x04, 0x01, 0x03, 0x01, 0x03  // port configuration block #3 (16/88): 4 max, 1/3 out, 1/3 in
0x04, 0x04, 0x01, 0x03, 0x01, 0x03  // port configuration block #4 (16/96): 4 max, 1/3 out, 1/3 in
0x05, 0x04, 0x01, 0x03, 0x01, 0x03  // port configuration block #5 (24/44): 4 max, 1/3 out, 1/3 in
0x06, 0x04, 0x01, 0x03, 0x01, 0x03  // port configuration block #6 (24/48): 4 max, 1/3 out, 1/3 in
0x0F                                // maximum length allowed for port name (15 ASCII characters)
0x04                                // length of port name field that follows (4 ASCII characters)
0x55, 0x53, 0x42, 0x31              // port name, the ASCII string “USB1”
0x01                                // port is first USB device jack
0x07                                // port supports PC/Mac and iOS audio, iOS audio is disabled
0xxx                                // checksum
0xF7                                // footer
```

4.5. GetAudioDeviceParm (Command ID = 0x44)

This command is used to query a device about details related to a specific audio device. Devices that support this command will respond with a RetAudioDeviceParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x44.
Data length is 2.

Bytes #1-2: audio port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x44                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // audio port ID
0xxx                            // checksum
0xF7                            // footer
```

4.6. RetAudioDeviceParm / SetAudioDeviceParm (Command ID = 0x45)

RetAudioDeviceParm is sent by devices in response to a GetAudioDeviceParm command.

SetAudioDeviceParm is sent by a host to a device to set the audio device details for USB host ports. Other types of audio ports cannot be modified using this command. Writeable parameters for USB host ports are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetAudioDeviceParm, Write for SetAudioDeviceParm.
Command ID is 0x45.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The host only needs to send up to (and including) the last writeable byte. All other bytes are read-only and are ignored.

Bytes #2-3: audio port ID (1 - N).
Byte #4: port type:

Value	Description
2	USB device
3	USB host
4	ethernet
5	analogue

Byte #5: maximum number of controllers supported by this device (0 - N).
Bytes #6-N: device detail, depends on port type (byte #4):

Type	Description
USB device	Byte #6: host type: 0 = no host, 1 = Mac/PC, 2 = iOS device Byte #7: length of host name field that follows (0 - N) Bytes #8-N: host name, 7-bit ASCII string, not NULL terminated Host name is only applicable for host type = iOS device. For Mac/PC hosts (or no host at all), length field is always 0x00 and host name is empty.

Type	Description
USB host	Byte #6: Flags: bit 7 = always zero bit 6 = set if a device is connected bits 5-1 = always zero bit 0 [W] = set if port is reserved for a specific vendor ID and product ID Byte #7: Maximum number of input channels supported by the hosted or reserved device, is 0 if no device is hosted or reserved on this port. Byte #8: Maximum number of output channels supported by the hosted or reserved device, is 0 if no device is hosted or reserved on this port. Byte #9: Number of output channel maps that follow (value should be the save as byte #5 from RetAudioPortParm. Bytes #N-N [W]: port output channel to device input channel map. Two bytes for each port output channel. First byte is port output channel number (1 - N), second byte is device input channel number (1 - N), or 0 if nothing is connected. There can only be one connection per device output channel. Byte #N: Number of input channel maps that follow (value should be the save as byte #6 from RetAudioPortParm. Bytes #N-N [W]: port input channel to device output channel map. Two bytes for each port input channel. First byte is port input channel number (1 - N), second byte is device output channel number (1 - N), or 0 if nothing is connected. There can only be one connection per device output channel. Bytes #N-N: Hosted or reserved device's USB vendor ID (16 bit value encoded in 3 bytes), is 0 if no device is hosted or reserved on this port. Bytes #N-N: Hosted or reserved device's USB product ID (16 bit value encoded in 3 bytes, is 0 if no device is hosted or reserved on this port). Byte #N: Length of vendor name field that follows (0 - N). Bytes #N-N: Vendor name, 7-bit ASCII string, not NULL terminated (from USB descriptor of hosted device). Byte #N: Length of product name field that follows (0 - N). Bytes #N-N: Product name, 7-bit ASCII string, not NULL terminated (from USB descriptor of hosted device).
ethernet	no details are available for ethernet ports at this time
analogue	no details are available for analogue ports at this time

Example 1 (USB device):

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x45                      // command flags and ID
0x00, 0x0B                      // data length (11)
0x01                            // command version number (1)
0x00, 0x01                      // port ID
0x02                            // port type = USB device
0x00                            // device has no controllers
```

```
0x02                        // host type = iOS device
0x04                        // length of ASCII string that follows (4)
0x69, 0x50, 0x61, 0x64      // host name, the ASCII string “iPad”
0xxx                        // checksum
0xF7                        // footer
```

Example 2 (USB host):

```
0xF0, 0x00, 0x01, 0x73, 0x7E  // header
0x00, 0x05                    // product ID
0x01, 0x02, 0x03, 0x04, 0x05  // serial number
0x00, 0x00                    // transaction ID
0x00, 0x45                    // command flags and ID
0x00, 0x2A                    // data length (42)
0x01                          // command version number (1)
0x00, 0x04                    // port ID
0x03                          // port type = USB host
0x02                          // device has 2 controllers
0x01                          // device is reserved but not connected
0x03                          // maximum # of input channels device supports (3)
0x02                          // maximum # of output channels device supports (2)
0x04                          // port has 4 outputs:
0x01, 0x02                    // port output #1 connected to device input #2
0x02, 0x01                    // port output #2 connected to device input #1
0x03, 0x03                    // port output #3 connected to device input #3
0x04, 0x00                    // port output #4 not connected to anything
0x04                          // port has 4 inputs:
0x01, 0x00                    // port input #1 not connected to anything
0x02, 0x00                    // port input #2 not connected to anything
0x03, 0x01                    // port input #3 connected to device output #1
0x04, 0x02                    // port input #4 connected to device output #2
0x00, 0x46, 0x21              // vendor ID (0x2321)
0x00, 0x00, 0x0F              // product ID (0x000F)
0x04                          // length of vendor name string that follows (4)
0x49, 0x43, 0x4F, 0x4E        // the ASCII string “ICON”
0x04                          // length of product name string that follows (4)
0x49, 0x43, 0x41, 0x34        // the ASCII string “ICA4”
0xxx                          // checksum
0xF7                          // footer
```

4.7. GetAudioControlParm (Command ID = 0x46)

This command is used to query a device about a controller for a specific audio device. Devices that support this command will respond with a RetAudioControlParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x46.
Data length is 3.

Bytes #1-2: audio port ID (1 - N).
Byte #3: controller number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x46                      // command flags and ID
0x00, 0x03                      // data length
0x00, 0x01                      // audio port ID
0x01                            // controller number
0xxx                            // checksum
0xF7                            // footer
```

4.8. RetAudioControlParm / SetAudioControlParm (Command ID = 0x47)

RetAudioControlParm is sent by devices in response to a GetAudioControlParm command.

SetAudioControlParm is sent by a host to a device to set the values for some of the controller parameters. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetAudioControlParm, Write for SetAudioControlParm.
Command ID is 0x47.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The host only needs to send up to (and including) the last writeable byte. All other bytes are read-only and are ignored.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: controller number (1 - N).
Byte #5: controller type:

Value	Description
5	Selector
6	Feature

Value	Description
10	Clock Source

Bytes #6-N: controller values, depends on controller type (byte #5):

Type	Description
Selector	Byte #6 [W]: current selector input Byte #7: number of selector inputs (controller details) Byte #8: length of controller name field that follows (0 - N) Bytes #9-N: controller name, 7-bit ASCII string, not NULL terminated
Feature	Byte #6: number of feature channels (controller details) Byte #7: length of controller name field that follows (0 - N) Bytes #8-N: controller name, 7-bit ASCII string, not NULL terminated
Clock Source	Byte #6 [W]: current clock source input Byte #7: number of clock source inputs (controller details) Byte #8: length of controller name field that follows (0 - N) Bytes #9-N: controller name, 7-bit ASCII string, not NULL terminated

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x47                      // command flags and ID
0x00, 0x0D                      // data length (13)
0x01                            // command version number (1)
0x00, 0x01                      // port ID
0x02                            // controller number (2)
0x05                            // controller type (selector)
0x02                            // current selector input (2)
0x03                            // number of selector inputs (3)
0x05                            // length of ASCII string that follows (5)
0x49, 0x6E, 0x70, 0x75, 0x74    // controller name, the ASCII string “Input”
0xxx                            // checksum
0xF7                            // footer
```

4.9. GetAudioControlDetail (Command ID = 0x48)

This command is used to query a device about the item details for a controller for a specific audio device. Devices that support this command will respond with a RetAudioControlDetail message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x48.
Data length is 4.

Bytes #1-2: audio port ID (1 - N).
Byte #3: controller number (1 - N).
Byte #4: detail number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x48                      // command flags and ID
0x00, 0x04                      // data length
0x00, 0x01                      // audio port ID
0x01                            // controller number
0x02                            // detail number
0xxx                            // checksum
0xF7                            // footer
```

4.10. RetAudioControlDetail (Command ID = 0x49)

RetAudioControlDetail is sent by devices in response to a GetAudioControlDetail command.

Command Data

Command flags are Answer.
Command ID is 0x49.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: controller number (1 - N).
Byte #5: detail number (1 - N).
Byte #6: controller type:

Value	Description
5	Selector
6	Feature
10	Clock Source

Bytes #7-N: controller detail, depends on controller type (byte #6):

Type	Description
Selector	Byte #7: length of selector input name field that follows (0 - N) Bytes #8-N: selector input name, 7-bit ASCII string, not NULL terminated

Type	Description
Feature	Byte #7: associated audio channel type: 0 = no association 1 = source (input) channel 2 = destination (output) channel Byte #8: associated audio channel number (1 - N), 0 if not associated. Byte #9: exist flags: bits 7-5 = always zero bit 4 = set if stereo link control exists bit 3 = set if high impedance control exists bit 2 = set if phantom power control exists bit 1 = set if mute control exists bit 0 = set if volume control exists Byte #10: edit flags: bits 7-5 = always zero bit 4 = set if stereo link control is editable bit 3 = set if high impedance control is editable bit 2 = set if phantom power control is editable bit 1 = set if mute control is editable bit 0 = set if volume control is editable The following 6 values are only present if volume control exists: Bytes #11-13: minimum value of volume control (16 bit value encoded in 3 bytes) Bytes #14-16: maximum value of volume control (16 bit value encoded in 3 bytes) Bytes #17-19: resolution of volume control (16 bit value encoded in 3 bytes) Bytes #20-22: pad value for volume control (16 bit value encoded in 3 bytes), optional offset applied to volume control values when using line inputs (as opposed to mic inputs), 0 means “no pad” Bytes #23-25: minimum value of trim control (16 bit value encoded in 3 bytes) Bytes #26-28: maximum value of trim control (16 bit value encoded in 3 bytes) Byte #29: length of channel name field that follows (0 - N) Bytes #30-N: channel name, 7-bit ASCII string, not NULL terminated The volume, trim, and pad values are 16 bit 2’s complement numbers (8.8 format) that can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001). Resolution can only have positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF). In addition, code 0x8000, representing silence (i.e., -∞ dB), must always be implemented. However, it must never be reported as the minimum value.
Clock Source	Byte #7: length of clock source name field that follows (0 - N) Bytes #8-N: clock source name, 7-bit ASCII string, not NULL terminated

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x00, 0x49                     // command flags and ID
0x00, 0x21                     // data length (33)
0x01                           // command version number (1)
0x00, 0x01                     // port ID
0x01                           // controller number (1)
0x02                           // detail number (2)
0x06                           // controller type (feature)
0x02                           // detail is associated to an output channel (1)
0x01                           // detail is associated to output channel 1
0x03, 0x03                     // both mute and volume controls exist and are editable
0x00, 0x00, 0x00               // minimum value of volume control (0 dB = 0x0000)
0x02, 0x00, 0x00               // maximum value of volume control (16 dB = 0x1000)
0x00, 0x02, 0x00               // resolution of volume control (1 dB = 0x0100)
0x00, 0x00, 0x00               // pad value (no pad)
0x00, 0x00, 0x00               // minimum value of trim control (0 dB = 0x0000)
0x02, 0x00, 0x00               // maximum value of trim control (16 dB = 0x1000)
0x04                           // length of ASCII string that follows (4)
0x4C, 0x65, 0x66, 0x74         // channel name, the ASCII string “Left”
0xxx                           // checksum
0xF7                           // footer
```

4.11. GetAudioControlDetailValue (Command ID = 0x4A)

This command is used to query a device about the values for a controller for a specific audio device. Devices that support this command will respond with a RetAudioControlDetailValue message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x4A.
Data length is 4.

Bytes #1-2: audio port ID (1 - N).
Byte #3: controller number (1 - N).
Byte #4: detail number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x40, 0x4A                     // command flags and ID
0x00, 0x04                     // data length
0x00, 0x01                     // audio port ID
0x01                           // controller number
0x02                           // detail number
```

0xxx // checksum
0xF7 // footer

4.12. RetAudioControlDetailValue / SetAudioControlDetailValue (Command ID = 0x4B)

RetAudioControlDetailValue is sent by devices in response to a GetAudioControlDetailValue command.

SetAudioControlDetailValue is sent by a host to a device to set the values for some of the controller details. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetAudioControlDetailValue, Write for SetAudioControlDetailValue.
Command ID is 0x4B.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The host only needs to send up to (and including) the last writeable byte. All other bytes are read-only and are ignored.

Bytes #2-3: audio port ID (1 - N).
Byte #4: controller number (1 - N).
Byte #5: detail number (1 - N).
Byte #6: controller type:

Value	Description
5	Selector
6	Feature
10	Clock Source

Bytes #7-N: controller detail, depends on controller type (byte #6):

Type	Description
Selector	not supported at this time

Type	Description
Feature	Byte #7: value flags: bits 7-5 = always zero bit 4 = set if stereo link control value is included bit 3 = set if high impedance control value is included bit 2 = set if phantom power control value included bit 1 = set if mute control value is included bit 0 = set if volume & trim control values are included Values follow for each of the flags in byte #7 (volume/trim first, mute second, etc): Bytes #8-10 [W]: value of volume control (16 bit value encoded in 3 bytes) Bytes #11-13 [W]: value of trim control (16 bit value encoded in 3 bytes) Byte #14 [W]: value of mute control (0 = mute off, 1 = mute on) Byte #15 [W]: value of phantom power control (0 = phantom power off, 1 = phantom power on) Byte #16 [W]: value of high impedance control (0 = high impedance off, 1 = high impedance on) Byte #17 [W]: value of stereo link control (0 = link off, 1 = link on) The volume and trim values are 16 bit 2's complement numbers (8.8 format) that can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001).
Clock Source	not supported at this time

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x4B                      // command flags and ID
0x00, 0x0E                      // data length (14)
0x01                            // command version number (1)
0x00, 0x01                      // port ID
0x01                            // controller number (1)
0x02                            // detail number (2)
0x06                            // controller type (feature)
0x03                            // mute and volume/trim value are included
0x00, 0x08, 0x00                // volume control value (4 dB = 0x0400)
0x00, 0x00, 0x00                // trim control value (0 dB = 0x0000)
0x00                            // mute is off (0)
0xxx                            // checksum
0xF7                            // footer
```

4.13. GetAudioClockParm (Command ID = 0x4C)

This command is used to query a device about audio clock sources. Devices that support this command will respond with a RetAudioClockParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x4C.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x4C                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

4.14. RetAudioClockParm / SetAudioClockParm (Command ID = 0x4D)

RetAudioClockParm is sent by devices in response to a GetAudioClockParm command. It contains information regarding audio clock sources for the device.

SetAudioClockParm is sent by a host to a device to set the audio clock source. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored. The host only needs to send up to (and including) byte #2, all bytes beyond #2 are read-only and are ignored. Parameters in RED underline require the device to be reset before taking effect.

Command Data

Command flags are Answer for RetAudioClockParm, Write for SetAudioClockParm.
Command ID is 0x4D.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2 [W]: number of the active audio clock source block (1 - N).
Byte #3: number of audio clock source blocks that follow (0 - N).

Bytes #4-N: audio clock source blocks. Bytes 4 and up are used only if byte #3 is not zero. Four bytes are used for each audio clock source block as follows:

Byte	Description
1	audio clock source number (1 - N)
2	clock source type: 1 = internal clock 2 = audio port clock
3 - 4	For clock source type = internal clock: Byte 3: internal clock ID (1 - N) Byte 4: reserved (always 0) For clock source type = audio port clock: Byte 3: audio port ID (1 - N) Byte 4: reserved (always 0)

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x4D                      // command flags and ID
0x00, 0x0F                      // data length (15)
0x01                            // command version number (1)
0x02                            // active audio clock source block is #2
0x03                            // number of audio clock blocks that follow (3)
0x01, 0x01, 0x01, 0x00          // audio clock source block #1: internal clock #1
0x02, 0x02, 0x01, 0x00          // audio clock source block #2: USB audio port #1
0x03, 0x02, 0x02, 0x00          // audio clock source block #3: USB audio port #2
0xxx                            // checksum
0xF7                            // footer
```

4.15. GetAudioPatchbayParm (Command ID = 0x4E)

This command is used to query a device about audio port patchbay setup. Devices that support this command will respond with a RetAudioPatchbayParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x4E.
Data length is 2.

Bytes #1-2: audio port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x4E                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // audio port ID
0xxx                            // checksum
0xF7                            // footer
```

4.16. RetAudioPatchbayParm / SetAudioPatchbayParm (Command ID = 0x4F)

RetAudioPatchbayParm is sent by devices in response to a GetAudioPatchbayParm command. It contains the patchbay setup for a specific audio port.

SetAudioPatchbayParm is sent by a host to a device to configure the patchbay for a specific audio port.

Command Data

Command flags are Answer for RetAudioPatchbayParm, Write for SetAudioPatchbayParm.
Command ID is 0x4F.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).

Byte #4: number of patchbay configuration blocks that follow (0 - N).

Bytes #5-N: patchbay configurations blocks. Bytes 5 and up are used only if byte #4 is not zero. If byte #4 is not zero, there should be one patchbay configuration block for each destination (output) channel for this port. Four bytes are used for each port configuration block as follows:

Byte	Description
1	destination (output) channel number (1 - N)
2	source (input) channel number (1 - N)
3 - 4	source (input) port ID (1 - N)

All values in each patch configuration block must be valid for the source port ID, source channel number, and destination channel number. Use 0x00 for bytes #2-4 to indicate “no connection” (i.e. no source (input) is connected to the destination (output) channel).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x4F                      // command flags and ID
0x00, 0x14                      // data length (20)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID
0x04                            // number of patchbay configuration blocks that follow (4)
0x01, 0x03, 0x00, 0x02          // block #1: output #1 is connected to port #2 channel 3
0x02, 0x01, 0x00, 0x03          // block #2: output #2 is connected to port #3 channel 1
0x03, 0x02, 0x00, 0x03          // block #3: output #3 is connected to port #3 channel 2
0x04, 0x00, 0x00, 0x00          // block #4: output #4 is not connected to anything
0xxx                            // checksum
0xF7                            // footer
```

4.17. GetAudioChannelName (Command ID = 0x3C)

This command is used to query a device about the names of channels on a specific audio port. Devices that support this command will respond with a RetAudioChannelName message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x3C.
Data length is 4.

Bytes #1-2: audio port ID (1 - N)
Byte #3: audio channel number (1 - N)
Byte #4: channel type (1 = source/input, 2 = destination/output)

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x3C                      // command flags and ID
0x00, 0x04                      // data length
0x00, 0x01                      // audio port ID (1)
0x03                            // audio channel (3)
0x02                            // channel type (2 = destination/output)
0xxx                            // checksum
0xF7                            // footer
```

4.18. RetAudioChannelName / SetAudioChannelName (Command ID = 0x3D)

RetAudioChannelName is sent by devices in response to a GetAudioChannelName command.

SetAudioChannelName is sent by a host to a device to set the name of an audio channel on a specific audio port. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetAudioChannelName, Write for SetAudioChannelName.
Command ID is 0x3D.
Data length depends on length of channel name.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: audio channel number (1 - N).
Byte #5: channel type (1 = source/input, 2 = destination/output)
Byte #6: maximum length allowed for channel name, 0 if read-only (0 - 127).
Byte #7 [W]: length of channel name field that follows (0 - N).
Bytes #8-N [W]: channel name, 7-bit ASCII string, not NULL terminated. Name must follow the same rules as used for device name (see RetInfo command).

Example:

0xF0, 0x00, 0x01, 0x73, 0x7E	// header
0x00, 0x05	// product ID
0x01, 0x02, 0x03, 0x04, 0x05	// serial number
0x00, 0x00	// transaction ID
0x00, 0x3D	// command flags and ID
0x00, 0x0B	// data length (11)
0x01	// command version number (1)
0x00, 0x01	// audio port ID (1)
0x03	// audio channel number (3)
0x02	// channel type (2 = destination/output)
0x0F	// maximum length allowed for channel name (15 ASCII characters)
0x04	// length of ASCII string that follows (4)
0x4C, 0x65, 0x66, 0x74	// channel name, the ASCII string “Left”
0xxx	// checksum
0xF7	// footer

4.19. GetAudioPortMeterValue (Command ID = 0x3E)

This command is used to query a device for the most recent meter values for a specific audio port. Devices that support this command will respond with a RetAudioPortMeterValue message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x3E.
Data length is 3.

Bytes #1-2: audio port ID (1 - N)
Byte #3: bitmap indicating which meter values should be returned from the device:

Bit	Description
7 - 2	always zero
1	channel destinations (outputs)
0	channel sources (inputs)

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x3E                      // command flags and ID
0x00, 0x03                      // data length
0x00, 0x01                      // audio port ID (1)
0x03                            // return meter values for destinations and sources
0xxx                            // checksum
0xF7                            // footer
```

4.20. RetAudioPortMeterValue (Command ID = 0x3F)

RetAudioPortMeterValue is sent by devices in response to a GetAudioPortMeterValue command.

Command Data

Command flags are Answer.
Command ID is 0x3F.
Data length depends on the number of audio channels.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: number of audio meter blocks that follow.
Bytes #5-N: audio meter blocks:

Byte	Description
1	Bitmap indicating which meter values are in this block: bits 7 - 2: always zero bit 1: channel destinations (outputs) bit 0: channel sources (inputs)
2	Number of meter values that follow.
3 - 4 (channel 1), 5 - 6 (channel 2), 7 - 8 (channel 3), etc.	Meter values. Each meter value is a 14 bit big-endian number split across 2 bytes. 0 dB is 8192 (0x2000) and is the maximum value that can be returned (representing digital full scale). The first value is channel 1, the second value is channel 2, etc. To convert to decibels (dB) use the formula: dB = 20 log (value/8192) Note that the minimum meter value (1) corresponds to -78.27 dB but resolution drops to 1 dB when meter value is 8 (-60 dB).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x3F                      // command flags and ID
0x00, 0x14                      // data length (20)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID (1)
0x02                            // number of audio meter blocks that follow (2)
0x01                            // block #1: channel sources/inputs (0x01)
0x02                            // meter readings that follow (2)
0x00, 0x00                      // input channel 1: meter value = 0x0000
0x20, 0x00                      // input channel 2: meter value = 0x1000
0x02                            // block #2: channel destinations/outputs (0x02)
0x04                            // meter readings that follow (4)
0x00, 0x00                      // output channel 1: meter value = 0x0000
0x40, 0x00                      // output channel 2: meter value = 0x2000
0x1F, 0x7F                      // output channel 3: meter value = 0x0FFF
0x01, 0x7F                      // output channel 4: meter value = 0x00FF
0xxx                            // checksum
0xF7                            // footer
```

5. Audio Mixer Commands

The following commands are defined for protocol version = 1.

5.1. GetMixerParm (Command ID = 0x50)

This command is used to query a device about audio mixer configurations. Devices that support this command will respond with a RetMixerParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x50.
Data length is 1.

Byte #1: audio configuration number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x50                      // command flags and ID
0x00, 0x01                      // data length
0x01                            // audio configuration number (1) [16 bit, 44100]
0xxx                            // checksum
0xF7                            // footer
```

5.2. RetMixerParm / SetMixerParm (Command ID = 0x51)

RetMixerParm is sent by devices in response to a GetMixerParm command. It contains information regarding mixer configurations. A host should cache this information and use it for further communication with this device.

SetMixerParm is sent by a host to a device to set the current mixer configuration. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored. Parameters in RED underline require the device to be reset before taking effect.

Command Data

Command flags are Answer for RetMixerParm, Write for SetMixerParm.
Command ID is 0x51.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The host only needs to send up to (and including) byte #2, all bytes beyond #2 are read-only and are ignored.

Byte #2 [W]: number of the active mixer configuration block (1 - N). Must be valid for the active audio configuration.
Byte #3: audio configuration number (1 - N).
Byte #4: number of audio configuration mixer blocks that follow (1 - N) for this audio configuration.
Bytes #5-N: audio configuration mixer blocks. Three bytes are used for each block as follows:

Byte	Description
1	mixer configuration number (1 - N)
2	maximum number of mixer inputs (1 - N)
3	maximum number of mixer outputs (1 - N)

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x51                      // command flags and ID
0x00, 0x0D                      // data length (13)
0x01                            // command version number (1)
0x03                            // active mixer configuration block (3)
0x01                            // audio configuration number (1) [16 bit, 44100]
0x03                            // number of mixer configuration blocks that follow (3)
0x01, 0x0C, 0x08                // block #1: 12 inputs, 8 outputs
0x02, 0x0A, 0x0A                // block #2: 10 inputs, 10 outputs
0x03, 0x08, 0x0C                // block #3: 8 inputs, 12 outputs
0xxx                            // checksum
0xF7                            // footer
```

5.3. GetMixerPortParm (Command ID = 0x52)

This command is used to query a device about audio port mixer configurations. Devices that support this command will respond with a RetMixerPortParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x52.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x52                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

5.4. RetMixerPortParm / SetMixerPortParm (Command ID = 0x53)

RetMixerPortParm is sent by devices in response to a GetMixerPortParm command. It contains information regarding audio port mixer configurations. A host should cache this information and use it for further communication with this device.

SetMixerPortParm is sent by a host to a device to set the current audio port mixer configuration. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored. Parameters in RED underline require the device to be reset before taking effect.

Command Data

Command flags are Answer for RetMixerPortParm, Write for SetMixerPortParm.
Command ID is 0x53.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2: number of audio port mixer blocks that follow (1 - N).
Bytes #3-N: audio port mixer blocks. One block for each audio port.

Byte	Description
1 - 2	audio port ID (1 - N)
3 [W]	number of mixer inputs to use for this audio port (0 - N) The number of inputs must be less than or equal to the number of inputs for the active mixer configuration.
4 [W]	number of mixer outputs to use for this audio port (0 - N) The sum of all outputs for all ports must be less than or equal to the number of outputs for the active mixer configuration.

For command version number = 2:

Byte #2: number of audio port mixer blocks that follow (1 - N).
Bytes #3-N: audio port mixer blocks. One block for each audio port.

Byte	Description
1 - 2	audio port ID (1 - N)
3 [W]	number of mixer inputs to use for this audio port (0 - N) The number of inputs must be less than or equal to the number of inputs for the active mixer configuration.
4 [W]	number of mixer outputs to use for this audio port (0 - N) The sum of all outputs for all ports must be less than or equal to the number of outputs for the active mixer configuration.
5	minimum number of mixer inputs allowed for this audio port (0 – N)
6	maximum number of mixer inputs allowed for this audio port (0 – N)
7	minimum number of mixer outputs allowed for this audio port (0 – N)
8	maximum number of mixer outputs allowed for this audio port (0 – N)

Example (active mixer configuration has 8 inputs and 12 outputs):

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x53                      // command flags and ID
0x00, 0x1A                      // data length (26)
0x02                            // command version number (2)
0x03                            // number of audio port mixer blocks that follow (3)
0x00, 0x01, 0x08, 0x02          // audio port #1: 8 inputs, 2 outputs
0x00, 0x08, 0x00, 0x06          // audio port #1: in min = 0, in max = 8, out min = 0, out max = 6
0x00, 0x02, 0x08, 0x06          // audio port #2: 8 inputs, 6 outputs
0x00, 0x08, 0x00, 0x06          // audio port #2: in min = 0, in max = 8, out min = 0, out max = 6
0x00, 0x03, 0x08, 0x04          // audio port #3: 8 inputs, 4 outputs
0x00, 0x08, 0x00, 0x06          // audio port #3: in min = 0, in max = 8, out min = 0, out max = 6
0xxx                            // checksum
0xF7                            // footer
```

5.5. GetMixerInputParm (Command ID = 0x54)

This command is used to query a device about mixer input assignments for a specific audio port. Devices that support this command will respond with a RetMixerInputParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x54.
Data length is 3.

Bytes #1-2: audio port ID (1 - N).
Byte #3: mixer input number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x54                      // command flags and ID
0x00, 0x03                      // data length
0x00, 0x01                      // audio port ID (1)
0x01                            // mixer input number (1)
0xxx                            // checksum
0xF7                            // footer
```

5.6. RetMixerInputParm / SetMixerInputParm (Command ID = 0x55)

RetMixerInputParm is sent by devices in response to a GetMixerInputParm command.

SetMixerInputParm is sent by a host to a device to set the mixer input channel parameters for a specific audio port. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMixerInputParm, Write for SetMixerInputParm.
Command ID is 0x55.
Data length is 7.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: mixer input number (1 - N).
Bytes #5-6 [W]: audio port ID of audio source (1 - N). 0 if mixer input is not assigned.
Byte #7 [W]: channel number of audio source (1 - N). 0 if mixer input is not assigned.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
```

```
0x00, 0x00                  // transaction ID
0x00, 0x55                  // command flags and ID
0x00, 0x07                  // data length (7)
0x01                        // command version number (1)
0x00, 0x01                  // audio port ID (1)
0x01                        // mixer input number (1)
0x00, 0x03                  // mixer input is audio port 3
0x04                        // mixer input is channel 4 (of audio port 3)
0xxx                        // checksum
0xF7                        // footer
```

5.7. GetMixerOutputParm (Command ID = 0x56)

This command is used to query a device about mixer output assignments for a specific audio port. Devices that support this command will respond with a RetMixerOutputParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x56.
Data length is 3.

Bytes #1-2: audio port ID (1 - N).
Byte #3: mixer output number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x56                      // command flags and ID
0x00, 0x03                      // data length
0x00, 0x01                      // audio port ID (1)
0x01                            // mixer output number (1)
0xxx                            // checksum
0xF7                            // footer
```

5.8. RetMixerOutputParm / SetMixerOutputParm (Command ID = 0x57)

RetMixerOutputParm is sent by devices in response to a GetMixerOutputParm command.

SetMixerOutputParm is sent by a host to a device to set the mixer output channel parameters for a specific audio port. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMixerOutputParm, Write for SetMixerOutputParm.

Command ID is 0x57.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: mixer output number (1 - N)
Byte #5 [W]: number of mixer output assignments that follow (0 - N).
Bytes #6-N [W]: channel number of mixer output assignment on this audio port (1 - N).
Byte #N: maximum length allowed for mix name, 0 if read-only (0 - 127).
Byte #N [W]: length of mix name field that follows (0 - N).
Bytes #N-N [W]: mix name, 7-bit ASCII string, not NULL terminated. Name must follow the same rules as used for device name (see RetInfo command).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x57                      // command flags and ID
0x00, 0x0D                      // data length (13)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID (1)
0x01                            // mixer output number (1)
0x02                            // number of mixer assignments that follow (2)
0x04                            // mixer output 1 is sent to output channel 4 (of audio port 1)
0x06                            // mixer output 1 is sent to output channel 6 (of audio port 1)
0x0F                            // maximum length allowed for mix name (15 ASCII characters)
0x04                            // length of ASCII string that follows (4)
0x4C, 0x65, 0x66, 0x74          // mix name, the ASCII string “Left”
0xxx                            // checksum
0xF7                            // footer
```

5.9. GetMixerInputControl (Command ID = 0x58)

This command is used to query a device about mixer input controls for a specific audio port. Devices that support this command will respond with a RetMixerInputControl message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x58.
Data length is 2.

Bytes #1-2: audio port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x40, 0x58                     // command flags and ID
0x00, 0x02                     // data length
0x00, 0x01                     // audio port ID (1)
0xxx                           // checksum
0xF7                           // footer
```

5.10. RetMixerInputControl (Command ID = 0x59)

RetMixerInputControl is sent by devices in response to a GetMixerInputControl command.

Command Data

Command flags are Answer.
Command ID is 0x59.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).

Byte #4: exist flags:
- bit 7 = always zero
- bit 6 = set if pan control exists
- bit 5 = set if invert control exists
- bit 4 = set if stereo link control exists
- bit 3 = set if solo PFL control exists
- bit 2 = set if solo control exists
- bit 1 = set if mute control exists
- bit 0 = set if volume control exists

Byte #5: edit flags:
- bit 7 = always zero
- bit 6 = set if pan control is editable
- bit 5 = set if invert control is editable
- bit 4 = set if stereo link control is editable
- bit 3 = set if solo PFL control is editable
- bit 2 = set if solo control is editable
- bit 1 = set if mute control is editable
- bit 0 = set if volume control is editable

The following 3 values are only present if pan control exists:
Bytes #6-8: maximum value of pan control representing a signal panned fully right (a positive 16 bit 2's complement value encoded in 3 bytes), the negative of this value represents a signal panned fully left
Byte #9: number of pan curve laws that follow

Bytes #10-N: pan curve law (only included if byte #9 is not zero):

Value	Description
1	0 dB at center
2	-1.5 dB at center
3	-3 dB at center
4	-4.5 dB at center
5	-6 dB at center

The pan values are 16 bit 2's complement numbers that can range from +32767 (0x7FFF) down to -32767 (0x8001). Positive numbers are panning to the right. Negative numbers are panning to the left. 0 means fully centered. The maximum pan value representing a signal panned fully right is specified in bytes #6-8, the minimum pan value representing a signal panned fully left is the negative of this value.

The following 3 values are only present if volume control exists:
Bytes #11-13: minimum value of volume control (16 bit value encoded in 3 bytes)
Bytes #14-16: maximum value of volume control (16 bit value encoded in 3 bytes)
Bytes #17-19: resolution of volume control (16 bit value encoded in 3 bytes)

The volume values are 16 bit 2's complement numbers (8.8 format) that can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001). Resolution can only have positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF). In addition, code 0x8000, representing silence (i.e., -∞ dB), must always be implemented. However, it must never be reported as the minimum value.

Note that for mixer inputs the solo control is a pushbutton (similar to mute), whereas for mixer outputs the solo control is a fader (similar to volume).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x59                      // command flags and ID
0x00, 0x15                      // data length (21)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID (1)
0x7F                            // all controls exist
0x7F                            // all controls are editable
0x00, 0x00, 0x7F                // maximum value of pan control (127)
0x03                            // number of pan curve laws that follow (3)
0x01                            // pan curve law #1: 0 dB at center
0x03                            // pan curve law #2: -3 dB at center
0x05                            // pan curve law #3: -6 dB at center
```

```
0x02, 0x40, 0x00        // minimum value of volume control (-96 dB)
0x00, 0x14, 0x00        // maximum value of volume control (+6 dB)
0x00, 0x00, 0x40        // resolution value of volume control (0.25 dB)
0xxx                    // checksum
0xF7                    // footer
```

5.11. GetMixerOutputControl (Command ID = 0x5A)

This command is used to query a device about mixer output controls for a specific audio port. Devices that support this command will respond with a RetMixerOutputControl message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x5A.
Data length is 2.

Bytes #1-2: audio port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x5A                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // audio port ID (1)
0xxx                            // checksum
0xF7                            // footer
```

5.12. RetMixerOutputControl (Command ID = 0x5B)

RetMixerOutputControl is sent by devices in response to a GetMixerOutputControl command.

Command Data

Command flags are Answer.
Command ID is 0x5B.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).

Byte #4: exist flags:
  bit 7 = always zero

bit 6 = set if pan control exists
bit 5 = set if invert control exists
bit 4 = set if stereo link control exists
bit 3 = set if solo PFL control exists
bit 2 = set if solo control exists
bit 1 = set if mute control exists
bit 0 = set if volume control exists

Byte #5: edit flags:
bit 7 = always zero
bit 6 = set if pan control is editable
bit 5 = set if invert control is editable
bit 4 = set if stereo link control is editable
bit 3 = set if solo PFL control is editable
bit 2 = set if solo control is editable
bit 1 = set if mute control is editable
bit 0 = set if volume control is editable

The following 3 values are only present if pan control exists:
Bytes #6-8: maximum value of pan control representing a signal panned fully right (a positive 16 bit 2's complement value encoded in 3 bytes), the negative of this value represents a signal panned fully left
Byte #9: number of pan curve laws that follow
Bytes #10-N: pan curve law (only included if byte #9 is not zero):

Value	Description
1	0 dB at center
2	-1.5 dB at center
3	-3 dB at center
4	-4.5 dB at center
5	-6 dB at center

The pan values are 16 bit 2's complement numbers that can range from +32767 (0x7FFF) down to -32767 (0x8001). Positive numbers are panning to the right. Negative numbers are panning to the left. 0 means fully centered. The maximum pan value representing a signal panned fully right is specified in bytes #6-8, the minimum pan value representing a signal panned fully left is the negative of this value.

The following 3 values are only present if volume or solo control exists:
Bytes #11-13: minimum value of volume or solo control (16 bit value encoded in 3 bytes)
Bytes #14-16: maximum value of volume or solo control (16 bit value encoded in 3 bytes)
Bytes #17-19: resolution of volume or solo control (16 bit value encoded in 3 bytes)

The volume and solo values are 16 bit 2's complement numbers (8.8 format) that can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001). Resolution can only have positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF).

In addition, code 0x8000, representing silence (i.e., -∞ dB), must always be implemented. However, it must never be reported as the minimum value.

Note that for mixer inputs the solo control is a pushbutton (similar to mute), whereas for mixer outputs the solo control is a fader (similar to volume).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x5B                      // command flags and ID
0x00, 0x15                      // data length (21)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID (1)
0x7F                            // all controls exist
0x7F                            // all controls are editable
0x00, 0x00, 0x7F                // maximum value of pan control (127)
0x03                            // number of pan curve laws that follow (3)
0x01                            // pan curve law #1: 0 dB at center
0x03                            // pan curve law #2: -3 dB at center
0x05                            // pan curve law #3: -6 dB at center
0x02, 0x40, 0x00                // minimum value of volume/solo control (-96 dB)
0x00, 0x14, 0x00                // maximum value of volume/solo control (+6 dB)
0x00, 0x00, 0x40                // resolution of volume/solo control (0.25 dB)
0xxx                            // checksum
0xF7                            // footer
```

5.13. GetMixerInputControlValue (Command ID = 0x5C)

This command is used to query a device about mixer input control values for a specific audio port. Devices that support this command will respond with a RetMixerInputControlValue message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x5C.
Data length is 4.

Bytes #1-2: audio port ID (1 - N).
Byte #3: mixer output number (1 - N).
Byte #4: mixer input number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x5C                      // command flags and ID
```

```
0x00, 0x04          // data length
0x00, 0x01          // audio port ID (1)
0x02                // mixer output number (2)
0x01                // mixer input number (1)
0xxx                // checksum
0xF7                // footer
```

5.14. RetMixerInputControlValue / SetMixerInputControlValue (Command ID = 0x5D)

RetMixerInputControlValue is sent by devices in response to a GetMixerInputControlValue command.

SetMixerInputControlValue is sent by a host to a device to set the mixer input control values for a specific audio port. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMixerInputControlValue, Write for SetMixerInputControlValue.
Command ID is 0x5D.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: mixer output number (1 - N)
Byte #5: mixer input number (1 - N)

Byte #6: value flags:
  bit 7 = always zero
  bit 6 = set if pan control and pan curve law values are included
  bit 5 = set if invert control value is included
  bit 4 = set if stereo link control value is included
  bit 3 = set if solo PFL control value is included
  bit 2 = set if solo control value is included
  bit 1 = set if mute control value is included
  bit 0 = set if volume control value is included

Values follow for each of the flags in byte #6 (volume first, mute second, etc):

Bytes #7-9 [W]: value of volume control (16 bit value encoded in 3 bytes)
Byte #10 [W]: value of mute control (0 = mute off, 1 = mute on)
Byte #11 [W]: value of solo control (0 = solo off, 1 = solo on)
Byte #12 [W]: value of solo PFL control (0 = solo PFL off, 1 = solo PFL on)
Byte #13 [W]: value of stereo link control (0 = link off, 1 = link on)
Byte #14 [W]: value of invert control (0 = invert off, 1 = invert on)
Bytes #15-17 [W]: value of pan control (16 bit value encoded in 3 bytes)
Byte #18 [W]: pan curve law

The volume values are 16 bit 2's complement numbers (8.8 format) that can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001).

The pan values are 16 bit 2's complement numbers that can range from +32767 (0x7FFF) down to -32767 (0x8001). Positive numbers are panning to the right. Negative numbers are panning to the left. 0 means fully centered.

Note that for mixer inputs the solo control is a pushbutton (similar to mute), whereas for mixer outputs the solo control is a fader (similar to volume).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x5D                      // command flags and ID
0x00, 0x0F                      // data length (15)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID (1)
0x02                            // mixer output number (2)
0x01                            // mixer input number (1)
0x47                            // pan, solo, mute and volume are included
0x00, 0x08, 0x00                // volume control value (4 dB = 0x0400)
0x00                            // mute is off (0)
0x01                            // solo is on (1)
0x00, 0x00, 0x00                // pan control is centered (0x0000)
0x03                            // pan curve law (-3 dB at center)
0xxx                            // checksum
0xF7                            // footer
```

5.15. GetMixerOutputControlValue (Command ID = 0x5E)

This command is used to query a device about mixer output control values for a specific audio port. Devices that support this command will respond with a RetMixerOutputControlValue message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x5E.
Data length is 3.

Bytes #1-2: audio port ID (1 - N).
Byte #3: mixer output number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
```

```
0x00, 0x00          // transaction ID
0x40, 0x5E          // command flags and ID
0x00, 0x03          // data length
0x00, 0x01          // audio port ID (1)
0x01                // mixer output number (1)
0xxx                // checksum
0xF7                // footer
```

5.16. RetMixerOutputControlValue / SetMixerOutputControlValue (Command ID = 0x5F)

RetMixerOutputControlValue is sent by devices in response to a GetMixerOutputControlValue command.

SetMixerOutputControlValue is sent by a host to a device to set the mixer output control values for a specific audio port. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMixerOutputControlValue, Write for SetMixerOutputControlValue.
Command ID is 0x5F.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: mixer output number (1 - N)

Byte #5: value flags:
- bit 7 = always zero
- bit 6 = set if pan control and pan curve law values are included
- bit 5 = set if invert control value is included
- bit 4 = set if stereo link control value is included
- bit 3 = set if solo PFL control value is included
- bit 2 = set if solo control value is included
- bit 1 = set if mute control value is included
- bit 0 = set if volume control value is included

Values follow for each of the flags in byte #5 (volume first, mute second, etc):

Bytes #6-8 [W]: value of volume control (16 bit value encoded in 3 bytes)
Byte #9 [W]: value of mute control (0 = mute off, 1 = mute on)
Bytes #10-12 [W]: value of solo control (16 bit value encoded in 3 bytes)
Byte #13 [W]: value of solo PFL control (0 = solo PFL off, 1 = solo PFL on)
Byte #14 [W]: value of stereo link control (0 = link off, 1 = link on)
Byte #15 [W]: value of invert control (0 = invert off, 1 = invert on)
Bytes #16-18 [W]: value of pan control (16 bit value encoded in 3 bytes)
Byte #19 [W]: pan curve law

The volume and solo values are 16 bit 2's complement numbers (8.8 format) that can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001).

The pan values are 16 bit 2's complement numbers that can range from +32767 (0x7FFF) down to -32767 (0x8001). Positive numbers are panning to the right. Negative numbers are panning to the left. 0 means fully centered.

Note that for mixer inputs the solo control is a pushbutton (similar to mute), whereas for mixer outputs the solo control is a fader (similar to volume).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x5F                      // command flags and ID
0x00, 0x0E                      // data length (14)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID (1)
0x01                            // mixer output number (1)
0x37                            // invert, link, solo, mute and volume are included
0x00, 0x08, 0x00                // volume control value (4 dB = 0x0400)
0x00                            // mute is off (0)
0x03, 0x78, 0x00                // solo control value (-4 dB = 0xFC00)
0x00                            // link is off (0)
0x01                            // invert is on (1)
0xxx                            // checksum
0xF7                            // footer
```

5.17. GetMixerMeterValue (Command ID = 0x60)

This command is used to query a device for the most recent meter values for a specific mixer. Devices that support this command will respond with a RetMixerMeterValue message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x60.
Data length is 4.

Bytes #1-2: audio port ID (1 - N).
Byte #3: mixer output number (1 - N).
Byte #4: bitmap indicating which meter values should be returned from the device:

Bit	Description
7 - 2	always zero

Bit	Description
1	mixer output channel
0	mixer input channels

A mixer always has one output but the number of inputs may be 0, 1, or some other value. For mixers configured as stereo pairs it is necessary to send two of these commands: one for the left channel (N, where N is an odd number) and one for the right channel (N+1).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x60                      // command flags and ID
0x00, 0x04                      // data length
0x00, 0x01                      // audio port ID (1)
0x02                            // mixer output number (2)
0x03                            // return meter values for mixer inputs and mixer output
0xxx                            // checksum
0xF7                            // footer
```

5.18. RetMixerMeterValue (Command ID = 0x61)

RetMixerMeterValue is sent by devices in response to a GetMixerMeterValue command.

Command Data

Command flags are Answer.
Command ID is 0x61.
Data length depends on the number of mixer channels.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: mixer output number (1 - N).
Byte #5: number of audio meter blocks that follow.
Bytes #6-N: audio meter blocks:

Byte	Description
1	Bitmap indicating which meter values are in this block: bits 7 - 2: always zero bit 1: mixer output channel bit 0: mixer input channels
2	Number of meter values that follow.
3 - 4 (channel 1), 5 - 6 (channel 2), 7 - 8 (channel 3), etc.	Meter values. Each meter value is a 14 bit big-endian number split across 2 bytes. 0 dB is 8192 (0x2000). The maximum value is 16383 (0x3FFF) or +6 dB. For inputs, the first value is mixer input 1, the second value is mixer input 2, etc. For outputs there is only ever one value. To convert to decibels (dB) use the formula: dB = 20 log (value/8192) Note that the minimum meter value (1) corresponds to -78.27 dB but resolution drops to 1 dB when meter value is 8 (-60 dB).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x61                      // command flags and ID
0x00, 0x0F                      // data length (15)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID (1)
0x02                            // mixer output number (2)
0x02                            // number of audio meter blocks that follow (2)
0x01                            // block #1: mixer inputs (0x01)
0x02                            // meter readings that follow (2)
0x00, 0x00                      // mixer input 1: meter value = 0x0000
0x20, 0x00                      // mixer input 2: meter value = 0x1000
0x02                            // block #2: mixer output (0x02)
0x04                            // meter readings that follow (1)
0x02, 0x00                      // mixer output: meter value = 0x0100
0xxx                            // checksum
0xF7                            // footer
```

6. Automation Control Commands

The following commands are defined for protocol version = 1.

6.1. GetAutomationControl (Command ID = 0x62)

This command is used to get information regarding automation controls that the device supports. Devices that support this command will respond with a RetAutomationControl message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x62.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x62                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

6.2. RetAutomationControl (Command ID = 0x63)

RetAutomationControl is sent by devices in response to a GetAutomationControl command. It contains information that a host will need to use to communicate with this device for other automation control related messages. A host should cache this information and use it for further communication with this device.

Command Data

Command flags are Answer.
Command ID is 0x63.
Data length depends on the number of automation controls.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2-3: maximum number of automation controls supported by this device (0 - N).
Byte #4: number of automation control blocks that follow (0 - N).

Bytes #5-N: automation control blocks. Bytes 5 and up are used only if byte #4 is not zero. The configuration of each block is as follows:

Byte	Description
1	Size of this block (including this byte).
2	Block type: 1 = audio port control 2 = audio port mixer input 3 = audio port mixer output 4 = snapshot list 5 = mute group
3 - N	For block type = audio port control (1): Bytes 3-4: audio port ID (1 - N) Byte 5: controller number (1 – N) Byte 6: controller type: 6 = Feature (other controller types not supported) Byte 7: bitmap for audio control features that support automation: bit 6 = trim control bit 1 = mute control bit 0 = volume control all other bits = 0 For block type = audio port mixer input (2): Bytes 3-4: audio port ID (1 - N) Byte 5: bitmap for mixer input controls that support automation: bit 6 = pan control bit 5 = invert control bit 3 = solo PFL control bit 2 = solo control bit 1 = mute control bit 0 = volume control all other bits = 0 For block type = audio port mixer output (3): Bytes 3-4: audio port ID (1 - N) Byte 5: bitmap for mixer output controls that support automation: bit 6 = pan control bit 5 = invert control bit 3 = solo PFL control bit 2 = solo control bit 1 = mute control bit 0 = volume control all other bits = 0 For block type = snapshot list (4) there are no additional bytes after byte 2. The snapshot list types that can be assigned to automation controls are returned in a RetSnapshotGlobalParm message. For block type = mute group (5) there are no additional bytes after byte 2. The mute group IDs that can be assigned to automation controls are returned in a RetHardwareGlobalParm message.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x63                      // command flags and ID
0x00, 0x19                      // data length (25)
0x01                            // command version number (1)
0x00, 0x64                      // device supports 100 automation controls
0x05                            // number of automation control blocks that follow (5)
0x07                            // block #1 size (7)
0x01                            // block #1 type: audio port control (1)
0x00, 0x02                      // block #1: audio port ID (2)
0x01                            // block #1: audio control number (1)
0x06                            // block #1: controller type: Feature (6)
0x03                            // block #1: mute and volume controls can be automated
0x05                            // block #2 size (5)
0x02                            // block #2 type: audio port mixer input (2)
0x00, 0x02                      // block #2: audio port ID (2)
0x43                            // block #2: pan, mute, and volume controls can be automated
0x05                            // block #3 size (5)
0x03                            // block #3 type: audio port mixer output (3)
0x00, 0x02                      // block #3: audio port ID (2)
0x23                            // block #3: pan, mute, and volume controls can be automated
0x02                            // block #4 size (2)
0x04                            // block #4 type: snapshot list (4)
0x02                            // block #5 size (2)
0x05                            // block #5 type: mute group (5)
0xxx                            // checksum
0xF7                            // footer
```

6.3. GetAutomationControlDetail (Command ID = 0x64)

This command is used to get detailed information about a specific automation control. Devices that support this command will respond with a RetAutomationControlDetail message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x64.
Data length is 2.

Bytes #1-2: automation control ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
```

```
0x00, 0x00          // transaction ID
0x40, 0x64          // command flags and ID
0x00, 0x02          // data length (2)
0x00, 0x01          // automation control ID (1)
0xxx                // checksum
0xF7                // footer
```

6.4. RetAutomationControlDetail / SetAutomationControlDetail (Command ID = 0x65)

RetAutomationControlDetail is sent by devices in response to a GetAutomationControlDetail command.

SetAutomationControlDetail is sent by a host to a device to configure a specific automation control.

Command Data

Command flags are Answer for RetAutomationControlDetail, Write for SetAutomationControlDetail.
Command ID is 0x65.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: automation control ID (1 – N).
Bytes #4-6: automation control key. The key defines the MIDI event type, MIDI channel, and MIDI event used for automation control as follows:

Byte	Description
4	MIDI event type: 0x00 = none (unassigned) 0x08 = note off (3-byte MIDI message) 0x09 = note on (3-byte MIDI message) 0x0A = poly aftertouch (3-byte MIDI message) 0x0B = continuous controller (3-byte MIDI message) 0x0C = program change (2-byte MIDI message) 0x0D = mono aftertouch (2-byte MIDI message) 0x0E = pitch bend (3-byte MIDI message)
5	MIDI channel number (1 – 16).
6	MIDI event. This corresponds to the 2nd byte in a 3-byte MIDI message (note off key, note on key, poly aftertouch key, continuous controller number, pitch bend LSB). For 2-byte MIDI messages (program change, mono aftertouch) this byte is always 0x00.

Automation controls that are unassigned will have 0x00 values for all three bytes of the automation control key and no other bytes are returned in the RetAutomationControlDetail message. To unassign an

automation control, a host should set all three bytes of the automation control key to 0x00 values in a SetAutomationControlDetail message, all other values in the message will be ignored.

Byte #7: automation control enable. Set to 0x00 if automation control is disabled, non-zero if automation control is enabled. This is a convenient way to disable an automation control without having to unassign it. If a host needs to enable or disable an automation control, it need only send up to and including this byte in a SetAutomationControlDetail message (the automation control key in bytes #4-6 must be included but the values sent by the host will be ignored).

Byte #8: automation control detail block type:

Value	Description
1	audio port control
2	audio port mixer input
3	audio port mixer output
4	snapshot list
5	mute group

Bytes #9-N: automation control detail block, depends on block type (byte #8):

Type	Description
audio port control (1)	Bytes #9-10: audio port ID (1 – N) Byte #11: controller number (1 - N) Byte #12: detail number (1 - N) Byte #13: controller type (6 = Feature, other controller types not supported) Byte #14: bitmap for this control (only 1 bit can be set) bit 6 = trim control bit 1 = mute control bit 0 = volume control all other bits = 0 For trim control: Byte #15: MIDI minimum value (0-127) Byte #16: MIDI maximum value (0-127) Bytes #17-19: trim minimum value For mute control: Byte #15: MIDI on value (0-127) Byte #16: flags: bit 0 = invert all other bits = 0

Type	Description
	For volume control: Byte #15: MIDI minimum value (0-127) Byte #16: MIDI maximum value (0-127) Bytes #17-19: volume minimum value, must be <= maximum value (bytes 20-22) Bytes #20-22: volume maximum value, must be >= minimum value (bytes 17-19) Byte #23: flags: bit 0 = mute, MIDI minimum value mutes volume (for audio outputs only) all other bits = 0
audio port mixer input (2)	Bytes #9-10: audio port ID (1 – N) Byte #11: mixer output number (1 - N) Byte #12: mixer input number (1 - N) Byte #13: bitmap for this control (only 1 bit can be set) bit 6 = pan control bit 5 = invert control bit 3 = solo PFL control bit 2 = solo control bit 1 = mute control bit 0 = volume control all other bits = 0 For pan control: Byte #14: MIDI minimum value (0-127) Byte #15: MIDI maximum value (0-127) Bytes #16-18: pan maximum value For invert, solo PFL, solo, and mute control: Byte #14: MIDI on value (0-127) Byte #15: flags: bit 0 = invert all other bits = 0 For volume control: Byte #14: MIDI minimum value (0-127) Byte #15: MIDI maximum value (0-127) Bytes #16-18: volume minimum value, must be <= maximum value (bytes 19-21) Bytes #19-21: volume maximum value, must be >= minimum value (bytes 16-18) Byte #22: flags: bit 0 = mute, MIDI minimum value mutes volume all other bits = 0
audio port mixer output (3)	Bytes #9-10: audio port ID (1 – N) Byte #11: mixer output number (1 - N) Byte #12: bitmap for this control (only 1 bit can be set) bit 6 = pan control bit 5 = invert control bit 3 = solo PFL control bit 2 = solo control bit 1 = mute control

Type	Description
	bit 0 = volume control all other bits = 0 For pan control: Byte #13: MIDI minimum value (0-127) Byte #14: MIDI maximum value (0-127) Bytes #15-17: pan maximum value For invert, solo PFL, and mute control: Byte #13: MIDI on value (0-127) Byte #14: flags: bit 0 = invert all other bits = 0 For solo and volume control: Byte #13: MIDI minimum value (0-127) Byte #14: MIDI maximum value (0-127) Bytes #15-17: volume minimum value, must be <= maximum value (bytes 18-20) Bytes #18-20: volume maximum value, must be >= minimum value (bytes 15-17) Byte #21: flags: bit 0 = mute, MIDI minimum value mutes volume all other bits = 0
snapshot list (4)	Byte #9: snapshot type Byte #10: control type 1 = continuous (e.g. fader or pan) 2 = pushbutton For continuous control: Byte #11: MIDI minimum value (0-127) Byte #12: MIDI maximum value (0-127) Byte #13: snapshot list minimum value (1-N), must be <= maximum value (byte 14) Byte #14: snapshot list maximum value (1-N), must be >= minimum value (byte 13) For pushbutton control: Byte #11: MIDI on value (0-127) Byte #12: flags: bit 0 = invert all other bits = 0 Byte #13: snapshot list value 0: decrement to previous snapshot in list 1 – 126: snapshot list index to select 127: increment to next snapshot in list
mute group (5)	Byte #9: control type 1 = continuous (e.g. fader or pan) 2 = pushbutton For continuous control:

Type	Description
	Byte #10: MIDI minimum value (0-127) Byte #11: MIDI maximum value (0-127) Byte #12: mute group minimum value (0-N), must be <= maximum value (byte 13) Byte #13: mute group maximum value (0-N), must be >= minimum value (byte 12) For bytes 12 and 13, values 1 – N are used to activate a mute group and 0 is used to deactivate all mute groups. For pushbutton control: Byte #10: MIDI on value (0-127) Byte #11: flags: bit 0 = invert all other bits = 0 Byte #12: mute group ID (1 – N) to activate when MIDI on value (or greater) is received. All mute groups are deactivated when a value less than the MIDI on value is received.

MIDI minimum and *MIDI maximum* refer to the MIDI values that will be used to map (for example) a slider on a MIDI controller to headphone output level or to sequence through a snapshot list. These are the values that appear in the 3rd byte of a 3-byte MIDI message (note off velocity, note on velocity, poly aftertouch value, continuous controller value, pitch bend MSB) or the 2nd byte of a 2-byte MIDI message (program change number, mono aftertouch value). MIDI values below *MIDI minimum* will be treated as *MIDI minimum*. MIDI values greater than *MIDI maximum* will be treated as *MIDI maximum*. A control's orientation can be inverted by setting *MIDI minimum* to a larger value than *MIDI maximum* (e.g. moving a MIDI slider upwards can cause headphone volume to decrease).

MIDI on refers to the MIDI value that will be used to map (for example) a pushbutton on a MIDI controller to headphone mute control, or to select a specific snapshot in a snapshot list, or to activate a mute group. This is the value that appears in the 3rd byte of a 3-byte MIDI message (note off velocity, note on velocity, poly aftertouch value, continuous controller value, pitch bend MSB) or the 2nd byte of a 2-byte MIDI message (program change number, mono aftertouch value). For example, if *MIDI on* is 0x40 then values between 0x40 to 0x7F will enable headphone mute function and values between 0x00 and 0x3F will disable headphone mute function. An *invert* flag can be used to reverse the logic so that values between 0x00 and 0x3F will enable headphone mute and values between 0x40 and 0x7F will disable headphone mute. *MIDI on* and *invert* flag are used for all pushbutton type automation controls (mute, solo, solo PFL, invert). For snapshot lists, pushbutton controls can be used to select a specific snapshot in the list, or increment to the next snapshot in the list, or decrement to the previous snapshot in the list whenever the *MIDI on* value is received. For mute groups, pushbutton controls can be used to activate a mute group when the pushbutton is pressed and deactivate all mute groups when the pushbutton is released.

For trim controls, *trim minimum* value is a 16 bit signed value (8.8 format) encoded into 3 bytes and must be within the range of the trim control's minimum and maximum values (as returned in a RetAudioContolDetail message). Only the integer part of the *trim minimum* value is used (the most significant 8 bits). Trim automation can only be used if the volume control exists and is editable (as returned in a RetAudioControlDetail message).

For audio port controls, mute automation can only be used if the mute control exists and is editable (as returned in a RetAudioControlDetail message).

For pan controls, *pan maximum* value is a 16 bit signed value encoded into 3 bytes and must be within range of the pan control maximum value (as returned in a RetMixerInputControl or RetMixerOutputControl message).

For volume controls (and mixer output solo controls), *volume minimum* and *volume maximum* are 16 bit signed values (8.8 format) encoded into 3 bytes and must be within the range of the volume control's minimum and maximum values (as returned in a RetAudioContolDetail, RetMixerInputControl or RetMixerOutputControl message). Only the integer part of the *volume minimum* and *volume maximum* values are used (the most significant 8 bits). A *mute* flag can be used to mute the volume control when the minimum MIDI value is received. The *mute* flag cannot be used with audio inputs. For audio port controls, volume automation can only be used if the volume control exists and is editable (as returned in a RetAudioControlDetail message).

Automation controls allow "many to many" connections between MIDI controls and device functions. For example, a single slider on a MIDI controller can be used to cross fade between two sets of inputs on a mixer (channel 1 volume increases from -20 dB to 0 dB while channel 2 volume decreases from -3 dB to -12 dB). Or, two sliders on a MIDI controller can be used to control a single audio channel, one set to change headphone volume over a 60 dB range and the other set to change headphone volume over a 10 dB range.

Example 1, mod wheel MIDI channel 4 controls headphone volume, MIDI range limited to 0x20 to 0x60, headphone volume limited to -20 dB to 0 dB, do not mute headphone on minimum MIDI value:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x65                      // command flags and ID
0x00, 0x17                      // data length (23)
0x01                            // command version number (1)
0x00, 0x01                      // automation control ID (1)
0x0B, 0x03, 0x01                // continuous controller, MIDI channel 4, mod wheel CC = 0x01
0x01                            // automation enabled (1)
0x01                            // block type: audio port control (1)
0x00, 0x02                      // audio port ID (2)
0x01                            // controller number (1)
0x01                            // detail number (1)
0x06                            // controller type (Feature = 6)
0x01                            // bitmap (volume control = 0x01)
0x20                            // MIDI minimum
0x60                            // MIDI maximum
0x03, 0x58, 0x00                // volume minimum (-20 dB, 0xEC00 in 8.8 format)
0x00, 0x00, 0x00                // volume maximum (0 dB, 0x0000 in 8.8 format)
0x00                            // mute flag (0 = do not mute on minimum MIDI value)
0xxx                            // checksum
0xF7                            // footer
```

Example 2, disable automation control 1 (do not unassign):

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
```

```
0x40, 0x65                  // command flags and ID
0x00, 0x07                  // data length (7)
0x01                        // command version number (1)
0x00, 0x01                  // automation control ID (1)
0x00, 0x00, 0x00            // key can be set to any values
0x00                        // automation disabled (0)
0xxx                        // checksum
0xF7                        // footer
```

Example 3, unassign automation control 1:

```
0xF0, 0x00, 0x01, 0x73, 0x7E  // header
0x00, 0x05                    // product ID
0x01, 0x02, 0x03, 0x04, 0x05  // serial number
0x00, 0x00                    // transaction ID
0x40, 0x65                    // command flags and ID
0x00, 0x06                    // data length (6)
0x01                          // command version number (1)
0x00, 0x01                    // automation control ID (1)
0x00, 0x00, 0x00              // key must be set to all 0x00 values to unassign
0xxx                          // checksum
0xF7                          // footer
```

7. Advanced MIDI Processor (AMP) Commands

The following commands are defined for protocol version = 1.

7.1. GetAMPGlobalParm (Command ID = 0x72)

This command is used to get information regarding AMP global parameters. Devices that support this command will respond with a RetAMPGlobalParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x72.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E  // header
0x00, 0x05                    // product ID
0x01, 0x02, 0x03, 0x04, 0x05  // serial number
0x00, 0x00                    // transaction ID
0x40, 0x72                    // command flags and ID
0x00, 0x00                    // data length
0xxx                          // checksum
0xF7                          // footer
```

7.2. RetAMPGlobalParm (Command ID = 0x73)

RetAMPGlobalParm is sent by devices in response to a GetAMPGlobalParm command. It contains information that a host will need to use to communicate with this device for other AMP related messages. A host should cache this information and use it for further communication with this device.

Command Data

Command flags are Answer.
Command ID is 0x73.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: maximum number of algorithms supported by this device (0 - N).
Bytes #4-5: maximum number of operators supported by this device (0 - N).
Bytes #6-7: maximum number of custom route maps supported by this device (0 - N).
Bytes #8-9: maximum number of lookup tables supported by this device (0 - N).
Byte #10: maximum number of operators allowed per algorithm (0 - N).
Byte #11: maximum length allowed for algorithm name (0 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x73                      // command flags and ID
0x00, 0x0B                      // data length (11)
0x01                            // command version number (1)
0x00, 0x10                      // number of algorithms supported (16)
0x00, 0x20                      // number of operators supported (32)
0x00, 0x08                      // number of custom route maps (8)
0x00, 0x08                      // number of lookup tables (8)
0x08                            // number of operators allowed per algorithm (8)
0x0F                            // max length allowed for algorithm name (15)
0xxx                            // checksum
0xF7                            // footer
```

7.3. GetAMPAlgorithmParm (Command ID = 0x74)

This command is used to get information regarding a specific AMP algorithm. Devices that support this command will respond with a RetAMPAlgorithmParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x74.

Data length is 2.

Bytes #1-2: algorithm ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x74                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // algorithm ID (1)
0xxx                            // checksum
0xF7                            // footer
```

7.4. RetAMPAlgorithmParm / SetAMPAlgorithmParm (Command ID = 0x75)

RetAMPAlgorithmParm is sent by devices in response to a GetAMPAlgorithmParm command.

SetAMPAlgorithmParm is sent by a host to a device to set the parameters for a specific algorithm.

Command Data

Command flags are Answer for RetAMPAlgorithmParm, Write for SetAMPAlgorithmParm.
Command ID is 0x75.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: algorithm ID (1 - N).
Byte #4: Length of algorithm name field that follows (0 - N).
Bytes #5-N: algorithm name, 7-bit ASCII string, not NULL terminated. Name must follow the same rules as used for device name (see RetInfo command).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x75                      // command flags and ID
0x00, 0x08                      // data length (8)
0x01                            // command version number (1)
0x00, 0x01                      // algorithm ID (1)
0x04                            // length of ASCII string that follows (4)
0x54, 0x65, 0x73, 0x74          // algorithm name, the ASCII string “Test”
0xxx                            // checksum
```

0xF7 // footer

7.5. GetAMPOperatorParm (Command ID = 0x76)

This command is used to get information regarding a specific AMP operator. Devices that support this command will respond with a RetAMPOperatorParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x76.
Data length is 2.

Bytes #1-2: operator ID (1 - N).

Example:

0xF0, 0x00, 0x01, 0x73, 0x7E // header
0x00, 0x05 // product ID
0x01, 0x02, 0x03, 0x04, 0x05 // serial number
0x00, 0x00 // transaction ID
0x40, 0x76 // command flags and ID
0x00, 0x02 // data length
0x00, 0x01 // operator ID (1)
0xxx // checksum
0xF7 // footer

7.6. RetAMPOperatorParm / SetAMPOperatorParm (Command ID = 0x77)

RetAMPOperatorParm is sent by devices in response to a GetAMPOperatorParm command.

SetAMPOperatorParm is sent by a host to a device to set the parameters for a specific operator.

Operators have one input pin and three output pins (Y, N, M). The Y and N output pins are from the output of the Match function. The M output pin is from the output of the Modify function. The Match function input is always connected to the input pin. The Modify function input can be connected to the Y or N output pins. Operators are assigned to algorithms, up to a maximum of N operators per algorithm (as returned from the RetAMPGlobalParm command).

Command Data

Command flags are Answer for RetAMPOperatorParm, Write for SetAMPOperatorParm.
Command ID is 0x77.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: operator ID (1 - N).
Bytes #4-17: Operator Connection Block:

Byte	Description
1 - 2	Algorithm ID (1 – N) that this operator is assigned to. Use 0x0000 if this operator is not assigned to any algorithm.
3	Connection Flags: bits 7 - 2: always zero bit 1: set if the N output pin is connected to the Modify function input. bit 0: set if the Y output pin is connected to the Modify function input.
4 - 5	Input pin connection for this operator: 0: input pin is not connected to anything. 1 – N: input pin connects to operator output with this ID. 0x3FFF: input pin connects to algorithm Entry point.
6	Pin ID of the operator output that connects to the input of this operator (only used if bytes 4-5 are not 0 or 0x3FFF): 0: no operator output pin is connected to the input of this operator. 1: the Y output pin of an operator is connected to the input of this operator. 2: the N output pin of an operator is connected to the input of this operator. 3: the M output pin of an operator is connected to the input of this operator.
7	MIDI channel to use for incoming MIDI event when operator M output is connected to the input of this operator (only used if bytes 4-5 are a valid operator ID and byte 6 has the value 3). 1 = MIDI channel 1, 2 = MIDI channel 2 … 16 = MIDI channel 16. Use 0 if incoming MIDI event should use the same MIDI channel as the original event.
8	Output Order. Sets the order of MIDI events from Y, N, and M outputs to algorithm Exit point. 0: YNM (Y output first, N output second, M output third). 1: YMN (Y output first, M output second, N output third). 2: NYM (N output first, Y output second, M output third). 3: NMY (N output first, M output second, Y output third). 4: MYN (M output first, Y output second, N output third). 5: MNY (M output first, N output second, Y output third).
9 - 10	Y output pin connection of this operator to algorithm Exit point: 0: Y output pin does not connect to algorithm Exit point. 1 – N: Y output pin connects to algorithm Exit point via custom route map with this ID. 0x3FFF: Y output pin connects to algorithm Exit point via default route map.
11 - 12	N output pin connection of this operator to algorithm Exit point: 0: N output pin does not connect to algorithm Exit point. 1 – N: N output pin connects to algorithm Exit point via custom route map with this ID. 0x3FFF: N output pin connects to algorithm Exit point via default route map.
13 - 14	M output pin connection of this operator to algorithm Exit point: 0: M output pin does not connect to algorithm Exit point. 1 – N: M output pin connects to algorithm Exit point via custom route map with this ID.

Byte	Description
	0x3FFF: M output pin connects to algorithm Exit point via default route map.

Bytes #18-22: Match Function Header Block:

Byte	Description
1	Match Event Type. Bitmask indicating which MIDI event types to match. More than 1 bit can be set: bit 7: always 0 bit 6: Pitch Bend (0xEn, 3-byte MIDI event) bit 5: Mono Aftertouch (0xDn, 2-byte MIDI event) bit 4: Program Change (0xCn, 2-byte MIDI event) bit 3: Controller (0xBn, 3-byte MIDI event) bit 2: Poly Aftertouch (0xAn, 3-byte MIDI event) bit 1: Note On (0x9n, 3-byte MIDI event) bit 0: Note Off (0x8n, 3-byte MIDI event)
2 - 4	Match Channel: Bitmask indicating which MIDI channels to match (0x0001 = MIDI channel 1, 0x0002 = MIDI channel 2 … 0x8000 = MIDI channel 16). More than 1 bit can be set (16-bit value encoded in 3 bytes).
5	Match Header Flags: bits 7 - 1: always zero bit 0: set if Match function uses 1x16 mode, clear if Match function uses 2x8 mode.

Bytes #23-N: If the Match function uses 2x8 mode (see Match Header Flags above) then two instances of the Match function 2x8 block are present: the first block is for byte-2 in the MIDI event, the second block is for byte-3 in the MIDI event. If the Match function uses 1x16 mode then one instance of the Match function 1x16 block is present.

Match Function 2x8 Block:

Byte	Description
1	Match 2x8 Flags: bits 7 - 2: always zero bit 1: set if a lookup table is used. bit 0: set if Match function logic is inverted.
2 - 3	Lookup table ID (1 – N). Only used if bit-1 is set in Match 2x8 Flags. Set to 0x0000 if not used.
4	Lookup table value that is used for matching (0x00 – 0x7F). Only used if bit-1 is set in Match 2x8 Flags.

Byte	Description
5	Minimum value that is used for matching (0x00 – 0x7F). Must be less than or equal to the maximum value (byte #6). Only used if bit-1 is clear in Match 2x8 Flags.
6	Maximum value that is used for matching (0x00 – 0x7F). Must be greater than or equal to the minimum value (byte #5). Only used if bit-1 is clear in Match 2x8 Flags.

Match Function 1x16 block:

Byte	Description
1	Match 1x16 Flags: bits 7 - 3: always zero bit 2: set if byte-3 (MSB) in MIDI event is used for lookup table offset, clear if byte-2 (LSB) in MIDI event is used for lookup table offset. bit 1: set if a lookup table is used. bit 0: set if Match function logic is inverted.
2 - 3	Lookup table ID (1 – N). Only used if bit-1 is set in Match 1x16 Flags. Set to 0x0000 if not used.
4	Lookup table value that is used for matching (0x00 – 0x7F). Only used if bit-1 is set in Match 1x16 Flags.
5 - 6	Minimum value that is used for matching (0x0000 – 0x3FFF). Must be less than or equal to the maximum value (bytes #7-8). Only used if bit-1 is clear in Match 1x16 Flags.
7 - 8	Maximum value that is used for matching (0x0000 – 0x3FFF). Must be greater than or equal to the minimum value (bytes #5-6). Only used if bit-1 is clear in Match 1x16 Flags.

Bytes #N-N: Modify Function Header Block:

Byte	Description
1	Modify In Event Type. Bitmask indicating which MIDI event types to modify. More than 1 bit can be set: bit 7: always 0 bit 6: Pitch Bend (0xEn, 3-byte MIDI event) bit 5: Mono Aftertouch (0xDn, 2-byte MIDI event) bit 4: Program Change (0xCn, 2-byte MIDI event) bit 3: Controller (0xBn, 3-byte MIDI event) bit 2: Poly Aftertouch (0xAn, 3-byte MIDI event) bit 1: Note On (0x9n, 3-byte MIDI event) bit 0: Note Off (0x8n, 3-byte MIDI event)

Byte	Description
2 - 4	Modify In Channel: Bitmask indicating which MIDI channels to modify (0x0001 = MIDI channel 1, 0x0002 = MIDI channel 2 … 0x8000 = MIDI channel 16). More than 1 bit can be set (16-bit value encoded in 3 bytes).
5	Modify Out Event Type. Sets the output MIDI event type. Only 1 bit can be set. Use 0x00 to select the same event type as the input MIDI event. bit 7: always 0 bit 6: Pitch Bend (0xEn, 3-byte MIDI event) bit 5: Mono Aftertouch (0xDn, 2-byte MIDI event) bit 4: Program Change (0xCn, 2-byte MIDI event) bit 3: Controller (0xBn, 3-byte MIDI event) bit 2: Poly Aftertouch (0xAn, 3-byte MIDI event) bit 1: Note On (0x9n, 3-byte MIDI event) bit 0: Note Off (0x8n, 3-byte MIDI event)
6 - 8	Modify Out Channel: Bitmask indicating which MIDI channels to use for outgoing MIDI events (0x0001 = MIDI channel 1, 0x0002 = MIDI channel 2 … 0x8000 = MIDI channel 16). More than 1 bit can be set (16-bit value encoded in 3 bytes). Use 0 to select the same channel as the input MIDI event.
9	Modify Header Flags: bits 7 - 1: always zero bit 0: set if Modify function uses 1x16 mode, clear if Modify function uses 2x8 mode.

Bytes #N-N: If the Modify function uses 2x8 mode (see Modify Header Flags above) then two instances of the Modify function 2x8 block are present: the first block is for byte-2 in the MIDI event, the second block is for byte-3 in the MIDI event. If the Modify function uses 1x16 mode then one instance of the Modify function 1x16 block is present.

Modify Function 2x8 Block:

Byte	Description
1	Modify 2x8 Flags: bits 7 - 3: always zero bit 2: set if Modify Default Value should be used when input is a 2-byte MIDI event. bit 1: set if this output byte uses input byte 3, clear if this output byte uses input byte 2. bit 0: set if a lookup table is used.
2 - 3	Lookup table ID (1 – N). Only used if bit-0 is set in Modify 2x8 Flags. Set to 0x0000 if not used.
4	Minimum in value that is used by Modify function (0x00 – 0x7F). Must be less than or equal to the maximum in value (byte #5).
5	Maximum in value that is used by Modify function (0x00 – 0x7F). Must be greater than or equal to the minimum in value (byte #4).

Byte	Description
6	Minimum out value that is used by Modify function (0x00 – 0x7F). Can be greater than maximum out value (byte #7) to allow for inversion.
7	Maximum out value that is used by Modify function (0x00 – 0x7F). Can be less than minimum out value (byte #6) to allow for inversion.
8	Minimum Default Mode. Selects the processing behavior for when the input byte value is less than Minimum In: 0: output value is set to input value, lookup table is not used. 1: output value is set to input value, lookup table is used if enabled. 2: output value is set to Minimum In value, lookup table is used if enabled. 3: output value is set to Minimum Default Value (byte #9), lookup table is not used.
9	Minimum Default Value, value to use when Minimum Default Mode (byte #8) is set to 0x03 (0x00 – 0x7F).
10	Maximum Default Mode. Selects the processing behavior for when the input byte value is greater than Maximum In: 0: output value is set to input value, lookup table is not used. 1: output value is set to input value, lookup table is used if enabled. 2: output value is set to Maximum In value, lookup table is used if enabled. 3: output value is set to Maximum Default Value (byte #11), lookup table is not used.
11	Maximum Default Value, value to use when Maximum Default Mode (byte #10) is set to 0x03 (0x00 – 0x7F).
12	Modify Default Value: Value to use when output is a 3-byte MIDI event but input is a 2-byte MIDI event (0x00 – 0x7F).

Modify function 1x16 block:

Byte	Description
1	Modify 1x16 Flags: bit 7: always zero. bit 6: set if MSB of the result is used for byte-2, clear if LSB of the result is used for byte-2 (only used when output is 2-byte MIDI event). bit 5: set if byte-2 is used for MSB and 0x00 is used for LSB, clear if byte-2 is used for LSB and 0x00 is used for MSB (only used when input is 2-byte MIDI event). bit 4: set if using a fixed value for byte-3 (MSB) rather than a lookup table value. bit 3: set if using a fixed value for byte-2 (LSB) rather than a lookup table value. bit 2: set if input byte-3 (MSB) value is used, clear if 0x00 is used for input byte-3. bit 1: set if input byte-2 (LSB) value is used, clear if 0x00 is used for input byte-2. bit 0: set if a lookup table is used.
2 - 3	Lookup table ID (1 – N) for byte-2 (LSB). Only used if bit-0 is set and bit-3 is clear in Modify 1x16 Flags. Set to 0x0000 if not used.

Byte	Description
4 - 5	Lookup table ID (1 – N) for byte-3 (MSB). Only used if bit-0 is set and bit-4 is clear in Modify 1x16 Flags. Set to 0x0000 if not used.
6	Byte-2 (LSB) fixed value to use instead of lookup table value (0x00 – 0x7F). Only used if bit-0 and bit-3 are both set in Modify 1x16 Flags.
7	Byte-3 (MSB) fixed value to use instead of lookup table value (0x00 – 0x7F). Only used if bit-0 and bit-4 are both set in Modify 1x16 Flags.
8 - 9	Minimum in value that is used by Modify function (0x0000 – 0x3FFF). Must be less than or equal to the maximum in value (bytes #10-11).
10 - 11	Maximum in value that is used by Modify function (0x0000 – 0x3FFF). Must be greater than or equal to the minimum in value (bytes #8-9).
12 - 13	Minimum out value that is used by Modify function (0x0000 – 0x3FFF). Can be greater than maximum out value (bytes #14-15) to allow for inversion.
14 - 15	Maximum out value that is used by Modify function (0x0000 – 0x3FFF). Can be less than minimum out value (bytes #12-13) to allow for inversion.
16	Minimum Default Mode. Selects the processing behavior for when the input value is less than Minimum In: 0: output value is set to input value, lookup table is not used. 1: output value is set to input value, lookup table is used if enabled. 2: output value is set to Minimum in value, lookup table is used if enabled. 3: output value is set to Minimum Default Value (bytes #17-18), lookup table is not used.
17 - 18	Minimum Default Value, value to use when Minimum Default Mode (byte #16) is set to 0x03 (0x0000 – 0x3FFF).
19	Maximum Default Mode. Selects the processing behavior for when the input value is greater than Maximum In: 0: output value is set to input value, lookup table is not used. 1: output value is set to input value, lookup table is used if enabled. 2: output value is set to Maximum in value, lookup table is used if enabled. 3: output value is set to Maximum Default Value (bytes #20-21), lookup table is not used.
20 - 21	Maximum Default Value, value to use when Maximum Default Mode (byte #19) is set to 0x03 (0x0000 – 0x3FFF).

Example 1 (single operator algorithm, Match and Modify functions both use 2x8 mode):

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x77                      // command flags and ID
0x00, 0x43                      // data length (67)
0x01                            // command version number (1)
```

```
0x00, 0x01              // operator ID (1)

                        // Operator Connection Block (length = 14)
0x00, 0x01              // algorithm ID (1)
0x01                    // flags: Y output pin is connected to Modify input,
0x7F, 0x7F              // operator input is connected to algorithm Entry point (0x3FFF)
0x00                    // no operator output pin is connected to the input of this operator
0x00                    // MIDI channel for incoming event (not used)
0x00                    // output order is YNM
0x00, 0x00              // Y output pin does not connect to Exit point (0x0000)
0x00, 0x04              // N output pin connects to Exit point via custom route map #4
0x7F, 0x7F              // M output pin connects to Exit point via default route map (0x3FFF)

                        // Match Function Header Block (length = 5)
0x03                    // match event types: note on and note off
0x02, 0x01, 0x05        // match channels: 1, 3, 8, 16
0x00                    // flags: use 2x8 mode

                        // Match Function 2x8 Block for byte-2 (length = 6)
0x02                    // flags: a lookup table is used
0x00, 0x08              // lookup table ID (8)
0x40                    // lookup table value used for matching (64)
0x00                    // minimum value (not used)
0x00                    // maximum value (not used)

                        // Match Function 2x8 Block for byte-3 (length = 6)
0x01                    // flags: logic is inverted
0x00, 0x00              // lookup table ID (not used)
0x00                    // lookup table value used for matching (not used)
0x20                    // minimum value (32)
0x80                    // maximum value (96)

                        // Modify Function Header Block (length = 9)
0x08                    // modify in event types: controller
0x03, 0x7F, 0x7F        // modify in channels: all
0x00                    // modify out event type: same as input (0)
0x00, 0x00, 0x00        // modify out channel: same as input (0)
0x00                    // flags: use 2x8 mode

                        // Modify Function 2x8 Block for byte-2 (length = 12)
0x01                    // flags: a lookup table is used
0x00, 0x07              // lookup table ID (7)
0x00                    // minimum in value (0)
0x7F                    // maximum in value (127)
0x00                    // minimum out value (0)
0x7F                    // maximum out value (127)
0x00                    // minimum default mode (0)
0x00                    // minimum default value (not used)
0x00                    // maximum default mode (0)
0x00                    // maximum default value (not used)
0x00                    // modify default value (0)

                        // Modify Function 2x8 Block for byte-3 (length = 12)
```

```
0x02                        // flags: use input byte 3
0x00, 0x00                  // lookup table ID (not used)
0x0A                        // minimum in value (10)
0x64                        // maximum in value (100)
0x00                        // minimum out value (0)
0x7F                        // maximum out value (127)
0x03                        // minimum default mode (3)
0x05                        // minimum default value (5)
0x00                        // maximum default mode (2)
0x00                        // maximum default value (not used)
0x00                        // modify default value (0)

0xxx                        // checksum
0xF7                        // footer
```

Example 2 (multi-operator algorithm, Match and Modify functions both use 1x16 mode):

```
0xF0, 0x00, 0x01, 0x73, 0x7E  // header
0x00, 0x05                    // product ID
0x01, 0x02, 0x03, 0x04, 0x05  // serial number
0x00, 0x00                    // transaction ID
0x00, 0x77                    // command flags and ID
0x00, 0x3C                    // data length (60)
0x01                          // command version number (1)
0x00, 0x02                    // operator ID (2)

                              // Operator Connection Block (length = 14)
0x00, 0x02                    // algorithm ID (2)
0x02                          // flags: N output pin is connected to Modify input
0x00, 0x03                    // ID of operator that connects to this input pin of this operator (3)
0x03                          // M output pin of operator #3 is connected to the input of this operator
0x07                          // MIDI channel to use for MIDI event from operator 3 M output (7)
0x05                          // output order is MNY
0x00, 0x00                    // Y output pin does not connect to Exit point (0x0000)
0x00, 0x00                    // N output pin does not connect to Exit point (0x0000)
0x00, 0x00                    // M output pin does not connect to Exit point (0x0000)

                              // Match Function Header Block (length = 5)
0x40                          // match event types: pitch bend
0x03, 0x7F, 0x7F              // match channels: all
0x01                          // flags: use 1x16 mode

                              // Match Function 1x16 Block (length = 8)
0x01                          // flags: logic is inverted
0x00, 0x00                    // lookup table ID (not used)
0x00                          // lookup table value used for matching (not used)
0x00, 0x20                    // minimum value (0x1000)
0x00, 0x60                    // maximum value (0x3000)

                              // Modify Function Header Block (length = 9)
0x40                          // modify in event types: pitch bend
0x03, 0x7F, 0x7F              // modify in channels: all
0x00                          // modify out event type: same as input (0)
```

```
0x00, 0x00, 0x20        // modify out channel: 6
0x01                    // flags: use 1x16 mode

                        // Modify Function 1x16 Block (length = 21)
0x01                    // flags: a lookup table is used
0x00, 0x05              // lookup table ID for LSB (5)
0x00, 0x06              // lookup table ID for MSB (6)
0x00                    // byte-2 fixed value (not used)
0x00                    // byte-3 fixed value (not used)
0x00, 0x00              // minimum in value (0x0000)
0x7F, 0x7F              // maximum in value (0x3FFF)
0x00, 0x60              // minimum out value (0x3000)
0x00, 0x20              // maximum out value (0x1000)
0x02                    // minimum default mode (2)
0x00, 0x00              // minimum default value (not used)
0x02                    // maximum default mode (2)
0x00, 0x00              // maximum default value (not used)

0xxx                    // checksum
0xF7                    // footer
```

7.7. GetAMPCustomRoute (Command ID = 0x78)

This command is used to query a device about a specific custom route map. Devices that support this command will respond with a RetAMPCustomRoute message. If this command is not supported the device will respond with an ACK message (with error code).

Note that this command is identical to the GetMIDIPortRoute command but uses custom route ID in place of MIDI port ID.

Command Data

Command flags are Query.
Command ID is 0x78.
Data length is 2.

Bytes #1-2: custom route ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x78                      // command flags and ID
0x00, 0x02                      // data length (2)
0x00, 0x01                      // custom route ID (1)
0xxx                            // checksum
0xF7                            // footer
```

7.8. RetAMPCustomRoute / SetAMPCustomRoute (Command ID = 0x79)

RetAMPCustomRoute is sent by devices in response to a GetAMPCustomRoute command.

SetAMPCustomRoute is sent by a host to a device to configure a custom route map. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Note that this command is identical to the RetMIDIPortRoute/SetMIDIPortRoute command but uses custom route ID in place of MIDI port ID.

Command Data

Command flags are Answer for RetAMPCustomRoute, Write for SetAMPCustomRoute.
Command ID is 0x79.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: custom route ID (1 - N).
Bytes #4-N [W]: bitmap indicating routing to all ports. Bit 0 is port #1, bit 1 is port #2, etc. Least significant byte is first, most significant byte is last. The most significant 4 bits of each byte must be 0 which allows 4 ports to be specified per byte. Unused bits should be set to 0. The number of ports is given in the RetMIDIInfo command. The number of bytes (in the bitmap) is an even value so that the bitmap (when unpacked) is a multiple of 8 bits:

number of bytes (using INTEGER math) = (((number of ports - 1) / 8) + 1) x 2

Example: device has 20 ports, custom route #1 sends to ports 2, 3, 7, 11-14, 19, 20

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x00, 0x79                     // command flags and ID
0x00, 0x09                     // data length (9)
0x01                           // command version number (1)
0x00, 0x01                     // custom route ID (1)
0x06                           // routing for ports 4 through 1
0x04                           // routing for ports 8 through 5
0x0C                           // routing for ports 12 through 9
0x03                           // routing for ports 16 through 13
0x08                           // routing for ports 20 through 17
0x00                           // routing for ports 24 through 21 (padding)
0xxx                           // checksum
0xF7                           // footer
```

7.9. GetAMPLookupTable (Command ID = 0x7A)

This command is used to query a device about a specific lookup table. Devices that support this command will respond with a RetAMPLookupTable message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x7A.
Data length is 2.

Bytes #1-2: lookup table ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x7A                      // command flags and ID
0x00, 0x02                      // data length (2)
0x00, 0x01                      // lookup table ID (1)
0xxx                            // checksum
0xF7                            // footer
```

7.10. RetAMPLookupTable / SetAMPLookupTable (Command ID = 0x7B)

RetAMPLookupTable is sent by devices in response to a GetAMPLookupTable command.

SetAMPLookupTable is sent by a host to a device to configure a lookup table.

Lookup tables are always 128 bytes long with values ranging from 0x00 through 0x7F.

Command Data

Command flags are Answer for RetAMPLookupTable, Write for SetAMPLookupTable.
Command ID is 0x7A.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: lookup table ID (1 - N).
Bytes #4-131: lookup table contents. Byte #4 is value at address 0 in lookup table, byte #5 is value at address 1 in lookup table, byte #131 is value at address 128 in lookup table.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
```

```
0x00, 0x05                          // product ID
0x01, 0x02, 0x03, 0x04, 0x05        // serial number
0x00, 0x00                          // transaction ID
0x00, 0x7B                          // command flags and ID
0x01, 0x03                          // data length (131)
0x01                                // command version number (1)
0x00, 0x01                          // lookup table ID (1)
0x00, 0x02, 0x04, 0x08              // lookup table values for addresses 0x00, 0x01, 0x02, 0x03
0x0A, 0x0C, 0x0E, 0x10              // lookup table values for addresses 0x04, 0x05, 0x06, 0x07
[28 more rows of lookup table data omitted for brevity]
0x10, 0x0E, 0x0C, 0x0A              // lookup table values for addresses 0x78, 0x79, 0x7A, 0x7B
0x08, 0x04, 0x02, 0x00              // lookup table values for addresses 0x7C, 07D, 0x7E, 0x7F
0xxx                                // checksum
0xF7                                // footer
```

7.11. GetAMPPortInfo (Command ID = 0x7C)

This command is used to query a device about AMP configuration for a specific MIDI port. Devices that support this command will respond with a RetAMPPortInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x7C.
Data length is 2.

Bytes #1-2: MIDI port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x7C                      // command flags and ID
0x00, 0x02                      // data length (2)
0x00, 0x01                      // MIDI port ID (1)
0xxx                            // checksum
0xF7                            // footer
```

7.12. RetAMPPortInfo / SetAMPPortInfo (Command ID = 0x7D)

RetAMPPortInfo is sent by devices in response to a GetAMPPortInfo command.

SetAMPPortInfo is sent by a host to a device to configure AMP for a specific MIDI port.

Command Data

Command flags are Answer for RetAMPPortInfo, Write for SetAMPPortInfo.
Command ID is 0x7D.

Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: MIDI port ID (1 - N).
Byte #4: bitmap indicating AMP status on MIDI in/out ports. Use these bits to enable/disable AMP on the MIDI in/out ports without having to change algorithm assignments. The MIDI in/out port must have a valid algorithm assigned to it before AMP can be enabled on that port.

Bit	Description
7 - 2	always zero
1	AMP is enabled on MIDI output
0	AMP is enabled on MIDI input

Bytes #5-6: algorithm ID assigned to MIDI input (1 – N), use 0x0000 if no algorithm is assigned.
Bytes #7-8: algorithm ID assigned to MIDI output (1 – N), use 0x0000 if no algorithm is assigned.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x7D                      // command flags and ID
0x00, 0x08                      // data length (8)
0x01                            // command version number (1)
0x00, 0x01                      // MIDI port ID (1)
0x01                            // AMP is enabled on MIDI input but not on MIDI output
0x00, 0x01                      // algorithm ID assigned to MIDI input (1)
0x00, 0x02                      // algorithm ID assigned to MIDI output (2)
0xxx                            // checksum
0xF7                            // footer
```


8. Snapshot Commands

Snapshots are views of a particular set of device parameters (e.g. MIDI routing, audio routing, or mixer settings) that can be saved for instant recall. Scenes are a specific grouping of snapshots and thus can include several different types of device parameters (e.g. MIDI routing and audio routing). Snapshots and scenes can be ordered into a snapshot/scene list (i.e. a set list) and sequenced using MIDI controllers via the automation control commands. Snapshot and scene lists can also be sequenced using footswitches. Most of the commands for managing snapshots are also used for managing scenes. However, CreateSnapshot cannot be used for creating scenes; use the GetSnapshotParm and RetSnapshotParm/SetSnapshotParm commands for viewing and creating scenes.

The following commands are defined for protocol version = 1.

8.1. GetSnapshotGlobalParm (Command ID = 0x66)

This command is used to get information regarding snapshot global parameters. Devices that support this command will respond with a RetSnapshotGlobalParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x66.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x66                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

8.2. RetSnapshotGlobalParm (Command ID = 0x67)

RetSnapshotGlobalParm is sent by devices in response to a GetSnapshotGlobalParm command. It contains information that a host will need to use to communicate with this device for other snapshot related messages. A host should cache this information and use it for further communication with this device.

Command Data

Command flags are Answer.
Command ID is 0x67.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2: maximum length allowed for snapshot name (0 – N).
Byte #3: number of snapshot type blocks that follow (0 – N).

Snapshot Type Block:

Byte	Description
1	Snapshot type (see table below).
2	Maximum number of snapshots allowed for this snapshot type (1 – N).
3	Maximum length of snapshot list (snapshot list index) for this snapshot type (1 – N).

Snapshot Types:

Value	Description
1	MIDI patchbay
2	Audio patchbay (and mixer input/output connections if mixers are supported)
3	Audio control
4	Mixer control (if mixers are supported)
127	Scene

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x67                      // command flags and ID
0x00, 0x0C                      // data length (12)
0x01                            // command version number (1)
0x0F                            // maximum length allowed for snapshot name (15)
0x03                            // number of snapshot blocks that follow (3)
0x01                            // block #1: snapshot type: MIDI patchbay (1)
0x08                            // block #1: max snapshots  (8)
0x10                            // block #1: max length of snapshot list  (16)
0x02                            // block #2: snapshot type: audio patchbay (2)
0x04                            // block #2: max snapshots (4)
0x10                            // block #2: max length of snapshot list (16)
0x7F                            // block #3: snapshot type: scene (127)
```

```
0x08                        // block #3: max scenes (8)
0x08                        // block #3: max length of scene list (8)
0xxx                        // checksum
0xF7                        // footer
```

8.3. GetSnapshotParm (Command ID = 0x68)

This command is used to get information regarding a specific snapshot. Devices that support this command will respond with a RetSnapshotParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x68.
Data length is 2.

Byte #1: snapshot type (1 - N).
Byte #2: snapshot ID (1 - N). Must be a valid ID for this snapshot type.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x68                      // command flags and ID
0x00, 0x02                      // data length
0x01                            // snapshot type: MIDI patchbay (1)
0x02                            // snapshot ID (2)
0xxx                            // checksum
0xF7                            // footer
```

8.4. RetSnapshotParm / SetSnapshotParm (Command ID = 0x69)

RetSnapshotParm is sent by devices in response to a GetSnapshotParm command.

SetSnapshotParm is sent by a host to a device to set the parameters for a specific snapshot.

Command Data

Command flags are Answer for RetSnapshotParm, Write for SetSnapshotParm.
Command ID is 0x69.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2: snapshot type (1 - N).

Byte #3: snapshot ID (1 - N). Must be a valid ID for this snapshot type.
Byte #4: length of snapshot name field that follows (0 – N).
Bytes #5-N: snapshot name, 7-bit ASCII string, not NULL terminated. Name must follow the same rules as used for device name (see RetInfo command).

For snapshot type = 1 (MIDI patchbay):

Byte #N: number of MIDI patchbay snapshot blocks that follow (0 – N).

MIDI Patchbay Snapshot block:

Byte	Description
1	Block type: 1: snapshot enable bitmask for MIDI input ports 2: snapshot enable bitmask for MIDI output ports
2	Number of bytes in bitmask field that follows.
3 - N	Bitmap used to enable or disable the snapshot on a per-port basis. Set the appropriate bit to change the MIDI port routing when the snapshot is applied. Clear the appropriate bit to leave the MIDI port routing unchanged when the snapshot is applied. This bitmap is identical in format to that used for MIDI port routing (see RetMIDIPortRoute command). Bit 0 is port #1, bit 1 is port #2, etc. Least significant byte is first, most significant byte is last. The most significant 4 bits of each byte must be 0 which allows 4 ports to be specified per byte. Unused bits should be set to 0. The number of ports is given in the RetMIDIInfo command. The number of bytes (in the bitmap) is an even value so that the bitmap (when unpacked) is a multiple of 8 bits: number of bytes (using INTEGER math) = (((number of ports - 1) / 8) + 1) x 2

Example (MIDI patchbay where device has 20 MIDI ports):

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x69                      // command flags and ID
0x00, 0x19                      // data length (25)
0x01                            // command version number (1)
0x01                            // snapshot type: MIDI patchbay (1)
0x02                            // snapshot ID (2)
0x04                            // length of snapshot name field that follows (4)
0x54, 0x65, 0x73, 0x74          // snapshot name, the ASCII string “Test”
0x02                            // number of MIDI patchbay snapshot blocks that follow (2)
0x01                            // block #1: type = MIDI input (1)
0x06                            // block #1: bitmask length (6 bytes)
0x06                            // block #1: bitmask for input ports 4 through 1
0x04                            // block #1: bitmask for input ports 8 through 5
0x0C                            // block #1: bitmask for input ports 12 through 9
0x03                            // block #1: bitmask for input ports 16 through 13
```

```
0x08    // block #1: bitmask for input ports 20 through 17
0x00    // block #1: bitmask for input ports 24 through 21 (padding)
0x02    // block #2: type = MIDI output (2)
0x06    // block #2: bitmask length (6 bytes)
0x02    // block #2: bitmask for output ports 4 through 1
0x07    // block #2: bitmask for output ports 8 through 5
0x0F    // block #2: bitmask for output ports 12 through 9
0x0A    // block #2: bitmask for output ports 16 through 13
0x05    // block #2: bitmask for output ports 20 through 17
0x00    // block #2: bitmask for output ports 24 through 21 (padding)
0xxx    // checksum
0xF7    // footer
```

For snapshot type = 2 (audio patchbay):

Byte #N: number of audio patchbay snapshot blocks that follow (0 – N).

Audio Patchbay Snapshot block:

Byte	Description
1	Block type: 1: snapshot enable bitmask for audio port inputs 2: snapshot enable bitmask for audio port outputs 3: snapshot enable bitmask for audio mixer inputs 4: snapshot enable bitmask for audio mixer outputs
2 - 3	Audio port ID (1 – N).
4	Number of bytes in bitmask field that follows.
5 - N	Bitmask used to enable or disable the snapshot on a per-channel basis. Set the appropriate bit to change the audio channel routing when the snapshot is applied. Clear the appropriate bit to leave the audio channel routing unchanged when the snapshot is applied. Bit 0 is channel #1, bit 1 is channel #2, etc. Least significant byte is first, most significant byte is last. The most significant 4 bits of each byte must be 0 which allows 4 channels to be specified per byte. Unused bits should be set to 0. The number of audio channels for each block type is determined as follows: audio port input (1): byte #5 from RetAudioPortParm message. audio port output (2): byte #6 from RetAudioPortParm message. audio mixer input (3): byte #3 of audio port mixer block from RetMixerPortParm message. audio mixer output (4): byte #4 of audio port mixer block from RetMixerPortParm message. The number of bytes (in the bitmap) is an even value so that the bitmap (when unpacked) is a multiple of 8 bits. number of bytes (using INTEGER math) = (((number of channels - 1) / 8) + 1) x 2.

Example (audio patchbay for audio port #1 which has 4 inputs, 6 outputs, and a mixer with 8 inputs and 4 outputs):

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x00, 0x69                     // command flags and ID
0x00, 0x21                     // data length (33)
0x01                           // command version number (1)
0x02                           // snapshot type: audio patchbay (2)
0x03                           // snapshot ID (3)
0x04                           // length of snapshot name field that follows (4)
0x54, 0x65, 0x73, 0x74         // snapshot name, the ASCII string “Test”
0x04                           // number of audio patchbay snapshot blocks that follow (4)
0x01                           // block #1: type = audio port input (1)
0x00, 0x01                     // block #1: audio port ID (1)
0x02                           // block #1: bitmask length (2 bytes)
0x06                           // block #1: bitmask for input ports 4 through 1
0x00                           // block #1: bitmask for input ports 8 through 5 (padding)
0x02                           // block #2: type = audio port output (2)
0x00, 0x01                     // block #2: audio port ID (1)
0x02                           // block #2: bitmask length (2 bytes)
0x0F                           // block #2: bitmask for output ports 4 through 1
0x03                           // block #2: bitmask for output ports 6 through 5
0x03                           // block #3: type = audio mixer input (3)
0x00, 0x01                     // block #3: audio port ID (1)
0x02                           // block #3: bitmask length (2 bytes)
0x05                           // block #3: bitmask for mixer inputs 4 through 1
0x0A                           // block #3: bitmask for mixer inputs 8 through 5
0x04                           // block #4: type = audio mixer output (4)
0x00, 0x01                     // block #4: audio port ID (1)
0x02                           // block #4: bitmask length (2 bytes)
0x00                           // block #4: bitmask for mixer outputs 4 through 1
0x00                           // block #4: bitmask for mixer outputs 8 through 5 (padding)
0xxx                           // checksum
0xF7                           // footer
```

For snapshot type = 3 (audio control):

Byte #N: number of audio control snapshot blocks that follow (0 – N).

Audio Control Snapshot block:

Byte	Description
1 - 2	audio port ID (1 – N).
3	Number of audio port control detail snapshot blocks that follow (1 – N).

Audio port control detail snapshot block:

Byte	Description
1	controller number (1 – N)
2	Controller type: 5 = Selector (not supported at this time) 6 = Feature 10 = Clock Source (not supported at this time)
3 - N	For controller type = Feature Byte 3: number of bitmask bytes that follow. Bytes 4 – N: Bitmasks used to enable or disable the snapshot on a per-control basis. Set the appropriate bit to change the control setting when the snapshot is applied. Clear the appropriate bit to leave the control setting unchanged when the snapshot is applied. One byte per controller detail (first byte is detail #1, second byte is detail #2, etc.) The number of controller details for each controller is given in the RetAudioControlParm command. bits 7-4 = always zero bit 3 = set if high impedance control is affected when snapshot is applied bit 2 = set if phantom power control is affected when snapshot is applied bit 1 = set if mute control is affected when snapshot is applied bit 0 = set if volume and trim controls are affected when snapshot is applied The controls must exist and be editable (as returned in a RetAudioControlDetail message).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x69                      // command flags and ID
0x00, 0x18                      // data length (24)
0x01                            // command version number (1)
0x03                            // snapshot type: audio control (3)
0x02                            // snapshot ID (2)
0x04                            // length of snapshot name field that follows (4)
0x54, 0x65, 0x73, 0x74          // snapshot name, the ASCII string “Test”
0x01                            // number of audio control snapshot blocks that follow (1)

                                // audio control snapshot block #1 (length = 15)
0x00, 0x03                      // audio port ID (3)
0x02                            // number of audio port control detail snapshot blocks (2)
0x01                            // block #1: controller number (1)
0x06                            // block #1: controller type = feature (6)
0x02                            // block #1: number of bitmasks that follow (2)
```

```
0x03                    // block #1: detail #1 bitmask (mute and volume)
0x01                    // block #1: detail #2 bitmask (volume)
0x02                    // block #2: controller number (2)
0x06                    // block #2: controller type = feature (6)
0x04                    // block #2: number of bitmasks that follow (4)
0x03                    // block #2: detail #1 bitmask (mute and volume)
0x01                    // block #2: detail #2 bitmask (volume)
0x02                    // block #2: detail #3 bitmask (mute)
0x00                    // block #2: detail #4 bitmask (none)

0xxx                    // checksum
0xF7                    // footer
```

For snapshot type = 4 (mixer control):

Byte #N: number of mixer control snapshot blocks that follow (0 – N).

Mixer Control Snapshot block:

Byte	Description
1 - 2	audio port ID (1 – N).
3	Number of mixer inputs that are used for this audio port (0 – N). Value should be the same as byte #3 in RetMixerPortParm.
4	Number of mixer outputs that are used for this audio port (0 – N). Value should be the same as byte #4 in RetMixerPortParm.
5 – N	Bitmasks used to enable or disable the snapshot on a per-control basis for mixer inputs. Set the appropriate bit to change the control setting when the snapshot is applied. Clear the appropriate bit to leave the control setting unchanged when the snapshot is applied. One byte per mixer input (and per mixer output). For example, if the number of mixer inputs is 3 (byte #3 above) and the number of mixer outputs is 2 (byte #4 above) there will be 6 bytes as follow: byte #1: mixer input #1 for mixer output #1 byte #2: mixer input #2 for mixer output #1 byte #3: mixer input #3 for mixer output #1 byte #4: mixer input #1 for mixer output #2 byte #5: mixer input #2 for mixer output #2 byte #6: mixer input #3 for mixer output #2 Each byte uses the following bitmask: bit 7 = always zero bit 6 = set if pan control is affected when snapshot is applied bit 5 = set if invert control is affected when snapshot is applied bit 4 = set if stereo link control is affected when snapshot is applied bit 3 = set if solo PFL control is affected when snapshot is applied

Byte	Description
	bit 2 = set if solo control is affected when snapshot is applied bit 1 = set if mute control is affected when snapshot is applied bit 0 = set if volume control is affected when snapshot is applied The controls must exist and be editable (as returned in a RetMixerInputControl message).
N – N	Bitmasks used to enable or disable the snapshot on a per-control basis for mixer outputs. Set the appropriate bit to change the control setting when the snapshot is applied. Clear the appropriate bit to leave the control setting unchanged when the snapshot is applied. One byte per mixer output. For example, if the number of mixer outputs is 2 (byte #4 above) there will be 2 bytes as follow: byte #1: mixer output #1 byte #2: mixer output #2 Each byte uses the following bitmask: bit 7 = always zero bit 6 = set if pan control is affected when snapshot is applied bit 5 = set if invert control is affected when snapshot is applied bit 4 = set if stereo link control is affected when snapshot is applied bit 3 = set if solo PFL control is affected when snapshot is applied bit 2 = set if solo control is affected when snapshot is applied bit 1 = set if mute control is affected when snapshot is applied bit 0 = set if volume control is affected when snapshot is applied The controls must exist and be editable (as returned in a RetMixerOutputControl message).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x69                      // command flags and ID
0x00, 0x15                      // data length (21)
0x01                            // command version number (1)
0x04                            // snapshot type: mixer control (4)
0x02                            // snapshot ID (2)
0x04                            // length of snapshot name field that follows (4)
0x54, 0x65, 0x73, 0x74          // snapshot name, the ASCII string “Test”
0x01                            // number of mixer control snapshot blocks that follow (1)

                                // mixer control snapshot block #1 (length = 12)
0x00, 0x03                      // mixer port ID (3)
0x03                            // number of mixer inputs (3)
0x02                            // number of mixer outputs (2)
0x01                            // bitmask for mixer input #1 of mixer output #1 (volume)
```

```
0x03                    // bitmask for mixer input #2 of mixer output #1 (volume + mute)
0x41                    // bitmask for mixer input #3 of mixer output #1 (volume + pan)
0x00                    // bitmask for mixer input #1 of mixer output #2 (none)
0x02                    // bitmask for mixer input #2 of mixer output #2 (mute)
0x40                    // bitmask for mixer input #3 of mixer output #2 (pan)
0x02                    // bitmask for mixer output #1 (mute)
0x01                    // bitmask for mixer output #2 (volume)

0xxx                    // checksum
0xF7                    // footer
```

For snapshot type = 127 (scene):

Byte #N: number of scene snapshot blocks that follow (0 – N). One block is included for each of the supported snapshot types (excluding scenes). For example, if the device supports snapshots for MIDI patchbay, audio patchbay, and scenes (but nothing else), this value should be 2.

Scene Snapshot block:

Byte	Description
1	Snapshot type (1 – N). Cannot be 127 (scene).
2	Snapshot ID (0 – N). Must be a valid ID for the snapshot type. Use 0 to indicate that no snapshot is used for this type (e.g. no values are changed for this type when this scene is selected).

Example (device supports snapshots for MIDI patchbay, audio patchbay, and scenes only):

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x69                      // command flags and ID
0x00, 0x0D                      // data length (13)
0x01                            // command version number (1)
0x7F                            // snapshot type: scene (127)
0x02                            // snapshot ID (2)
0x04                            // length of snapshot name field that follows (4)
0x54, 0x65, 0x73, 0x74          // snapshot name, the ASCII string “Test”
0x02                            // number of scene snapshot blocks that follow (2)
0x01                            // block #1: snapshot type = MIDI patchbay (1)
0x03                            // block #1: snapshot ID (3)
0x02                            // block #2: snapshot type = audio patchbay (2)
0x00                            // block #2: snapshot ID (0 = none)
0xxx                            // checksum
0xF7                            // footer
```

8.5. GetSnapshotList (Command ID = 0x6A)

This command is used to get information regarding a specific snapshot list. Devices that support this command will respond with a RetSnapshotList message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x6A.
Data length is 1.

Byte #1: snapshot type (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x6A                      // command flags and ID
0x00, 0x01                      // data length
0x01                            // snapshot type: MIDI patchbay (1)
0xxx                            // checksum
0xF7                            // footer
```

8.6. RetSnapshotList / SetSnapshotList (Command ID = 0x6B)

ReSnapshotList is sent by devices in response to a GetSnapshotList command.

SetSnapshotList is sent by a host to a device to set the parameters for a specific snapshot list. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetSnapshotList, Write for SetSnapshotList.
Command ID is 0x6B.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2: snapshot type (1 - N).
Byte #3 [W]: snapshot list flags:

Bit	Description
7 - 1	always zero

Bit	Description
0	Loop enable. Set if looping through the snapshot list is enabled (incrementing above the top of the list will select the item at the bottom of the list, decrementing below the bottom of the list will select the item at the top of the list). Clear if looping is not enabled.

Byte #4: last applied snapshot ID (1 – N), 0 if no snapshot has been applied.
Byte #5: last applied snapshot list index (1 – N), 0 if no snapshot list index has been applied.
Byte #6 [W]: length of snapshot list that follows.
Bytes #7-N [W]: snapshot IDs (1 – N). Must be valid snapshot IDs for the snapshot type.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x6B                      // command flags and ID
0x00, 0x0D                      // data length (13)
0x01                            // command version number (1)
0x01                            // snapshot type: MIDI patchbay (1)
0x01                            // snapshot flags: loop enable (0x01)
0x03                            // last applied snapshot ID (3)
0x05                            // last applied snapshot list index (5)
0x07                            // length of snapshot list that follows (7)
0x02                            // snapshot list index 1: snapshot ID = 2
0x01                            // snapshot list index 2: snapshot ID = 1
0x02                            // snapshot list index 3: snapshot ID = 2
0x03                            // snapshot list index 4: snapshot ID = 3
0x02                            // snapshot list index 5: snapshot ID = 2
0x05                            // snapshot list index 6: snapshot ID = 5
0x02                            // snapshot list index 7: snapshot ID = 2
0xxx                            // checksum
0xF7                            // footer
```

8.7. CreateSnapshot (Command ID = 0x6C)

This command is used to create a snapshot. If the device supports GetSnapshotParm command then the device supports this command. If this command fails or is not supported the device will respond with an ACK message (with error code). To create scenes, first use the SetSnapshotParm command to configure which snapshots will be used in the scene, then use CreateSnapshot (using the SCENE snapshot type) to create all the snapshots that the scene uses.

Command Data

Command flags are Query.
Command ID is 0x6C.
Data length is 3.

Byte #1: command version number that this device supports (1 - 127). Same value as returned in RetSnapshotParm command.
Byte #2: snapshot type (1 - N).
Byte #3: snapshot ID (1 - N). Must be a valid ID for this snapshot type.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x6C                      // command flags and ID
0x00, 0x03                      // data length
0x01                            // command version number (1)
0x01                            // snapshot type: MIDI patchbay (1)
0x03                            // snapshot ID (3)
0xxx                            // checksum
0xF7                            // footer
```

8.8. ApplySnapshot (Command ID = 0x6D)

This command is used to apply a snapshot. If the device supports GetSnapshotParm command then the device supports this command. If this command fails or is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x6D.
Data length is 3.

Byte #1: command version number that this device supports (1 - 127). Same value as returned in RetSnapshotParm command.
Byte #2: snapshot type (1 - N).
Byte #3: snapshot ID (1 – N). Must be a valid snapshot ID for the snapshot type.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x6D                      // command flags and ID
0x00, 0x03                      // data length
0x01                            // command version number (1)
0x01                            // snapshot type: MIDI patchbay (1)
0x03                            // snapshot ID (3)
0xxx                            // checksum
0xF7                            // footer
```

8.9. ApplySnapshotList (Command ID = 0x6E)

This command is used to apply a snapshot via a snapshot list. If the device supports GetSnapshotList command then the device supports this command. If this command fails or is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x6E.
Data length is 3.

Byte #1: command version number that this device supports (1 - 127). Same value as returned in RetSnapshotList command.
Byte #2: snapshot type (1 – N).
Byte #3: snapshot list index (0 – N). Note the following rules:
- 0: decrement to previous snapshot in list
- 1 – 126: snapshot list index to select, must be a valid index for this snapshot list.
- 127: increment to next snapshot in list

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x6E                      // command flags and ID
0x00, 0x03                      // data length
0x01                            // command version number (1)
0x01                            // snapshot type: MIDI patchbay (1)
0x04                            // snapshot list index (4)
0xxx                            // checksum
0xF7                            // footer
```

9. Hardware Interface Commands

The following commands are defined for protocol version = 1.

9.1. GetHardwareGlobalParm (Command ID = 0x80)

This command is used to get information regarding which hardware interface elements a device supports. Devices that support this command will respond with a RetHardwareGlobalParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x80.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x41, 0x00                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

9.2. RetHardwareGlobalParm (Command ID = 0x81)

RetHardwareGlobalParm is sent by devices in response to a GetHardwareGlobalParm command. It contains information that a host will need to use to communicate with this device for other hardware interface related messages. A host should cache this information and use it for further communication with this device.

Command Data

Command flags are Answer.
Command ID is 0x81.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2: number of hardware interface type blocks that follow (0 – N).
Bytes #3-N: hardware interface type blocks:

Hardware Interface Type Block:

Byte	Description
1	Size of this block (including this byte).
2	Hardware Interface type: 1: Footswitch 2: Mute Group 3: Automatic Failover 4: Tone Generator
3 - N	Dependent on hardware interface type (see tables below).

For hardware interface type = 1 (footswitch):

Byte	Description
3	Number of footswitches supported by this device.
4	Number of footswitch functions supported by this device (number of bytes that follow).
5 - N	Footswitch function codes: 1: Snapshot Selection 2: Mute Group Selection 3: Automatic Failover Setting

For hardware interface type = 2 (mute group):

Byte	Description
3	Number of mute groups supported by this device.

For hardware interface type = 3 (automatic failover):

Byte	Description
	No additional bytes are used for this hardware interface type.

For hardware interface type = 4 (tone generator):

Byte	Description
	No additional bytes are used for this hardware interface type.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x01, 0x01                      // command flags and ID
0x00, 0x0F                      // data length (15)
0x01                            // command version number (1)
0x04                            // number of hardware interface type blocks that follow (4)
0x06                            // block #1: size of this block (6)
0x01                            // block #1: hardware interface type: footswitch (1)
0x04                            // block #1: number of footswitches (4)
0x02                            // block #1: number of footswitch functions that follow (2)
```

```
0x01                    // block #1: footswitch function #1 (snapshot selection) (1)
0x02                    // block #1: footswitch function #2 (mute group selection) (2)
0x03                    // block #2: size of this block (3)
0x02                    // block #2: hardware interface type: mute group (2)
0x03                    // block #2: number of mute groups (3)
0x02                    // block #3: size of this block (2)
0x03                    // block #3: hardware interface type: automatic failover (3)
0x02                    // block #4: size of this block (2)
0x04                    // block #4: hardware interface type: tone generator (4)
0xxx                    // checksum
0xF7                    // footer
```

9.3. GetHardwareParm (Command ID = 0x82)

This command is used to get information regarding a specific hardware interface element. Devices that support this command will respond with a RetHardwareParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x82.
Data length is 2.

Byte #1: hardware interface type (1 - N).

For hardware interface type = 1 (footswitch):

Byte #2: footswitch ID (1 – N). Must be a valid footswitch ID for this device.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x41, 0x02                      // command flags and ID
0x00, 0x02                      // data length
0x01                            // hardware interface type: footswitch (1)
0x04                            // footswitch ID (4)
0xxx                            // checksum
0xF7                            // footer
```

For hardware interface type = 2 (mute group):

Byte #2: mute group ID (1 – N). Must be a valid mute group ID for this device.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
```

```
0x00, 0x00                  // transaction ID
0x41, 0x02                  // command flags and ID
0x00, 0x02                  // data length
0x02                        // hardware interface type: mute group (2)
0x01                        // mute group ID (1)
0xxx                        // checksum
0xF7                        // footer
```

For hardware interface type = 3 (automatic failover):

Byte #2: always 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E  // header
0x00, 0x05                    // product ID
0x01, 0x02, 0x03, 0x04, 0x05  // serial number
0x00, 0x00                    // transaction ID
0x41, 0x02                    // command flags and ID
0x00, 0x02                    // data length
0x03                          // hardware interface type: automatic failover (3)
0x00                          // always 0
0xxx                          // checksum
0xF7                          // footer
```

For hardware interface type = 4 (tone generator):

Byte #2: always 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E  // header
0x00, 0x05                    // product ID
0x01, 0x02, 0x03, 0x04, 0x05  // serial number
0x00, 0x00                    // transaction ID
0x41, 0x02                    // command flags and ID
0x00, 0x02                    // data length
0x04                          // hardware interface type: tone generator (4)
0x00                          // always 0
0xxx                          // checksum
0xF7                          // footer
```

9.4. RetHardwareParm / SetHardwareParm (Command ID = 0x83)

RetHardwareParm is sent by devices in response to a GetHardwareParm command.

SetHardwareParm is sent by a host to a device to set the parameters for a specific hardware interface element. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetHardwareParm, Write for SetHardwareParm.
Command ID is 0x83.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2: hardware interface type (1 - N).

For hardware interface type = 1 (footswitch):

Footswitches support different types of jacks. Some jacks may have more than one connection (e.g. TRS phone jacks can support two footswitch functions on a single jack). Some devices may have footswitch jacks that only support inputs or outputs whereas another device may support both input and output on the same physical connection (user configurable to be either an input or an output). Footswitches can be either latching (e.g. guitar stomp box switch) or momentary (e.g. piano sustain peddle). Footswitches can be normally open or normally closed.

Footswitches can be used to control snapshot sequencing, mute operations, and other functions.

Byte #3: footswitch ID (1 - N). Must be a valid footswitch ID for this device.
Bytes #4 – N: footswitch block.

Footswitch block:

Byte	Description
4	Jack ID (1 – N)
5	Jack Type: 1: 1/4 inch phone jack 2: 1/8 inch phone jack
6	Location of this footswitch on this jack: For Jack Type = 1/4 inch phone jack and 1/8 inch phone jack 1: tip 2: ring
7	Flags: bit 7: reserved (always 0) bit 6 [W]: set if footswitch is enabled, clear if footswitch is disabled bit 5 [W]: set if footswitch is momentary, clear if footswitch is latching bit 4 [W]: set if footswitch is inverted (normally closed), clear if footswitch is not inverted (normally open) bit 3 [W]: set if footswitch state should be read and applied on power up (inputs only) bit 2 [W]: set if port is currently configured to be an input, clear if port is currently configured to be an output, value must agree with bits 1 and 0 bit 1: set if port supports output bit 0: set if port supports input

Byte	Description
8 - 9 [W]	Debounce time (inputs) or Hold time (outputs, if momentary flag is set) in milliseconds (1 – 6000). Can also use 0 to indicate “minimum time supported by device” which may be less than 1 ms.
10 [W]	Function code. Must be a valid function code supported by the device (see “footswitch function codes” in RetHardwareGlobalParm command).
11 - N	Depends on function code (see tables below).

For function code = 1 (snapshot selection):

Footswitch Snapshot Selection block:

Byte	Description
11 [W]	Snapshot type (1 – N). Must be a supported snapshot type for the device.
12	Flags: bits 7 - 2: reserved (always 0) bit 1 [W]: set if inactive state (byte 14) refers to snapshot list index, clear if inactive state refers to snapshot ID. bit 0 [W]: set if active state (byte 13) refers to snapshot list index, clear if active state index refers to snapshot ID.
13 [W]	Active state snapshot ID or snapshot list index (0 – N). Must be a valid value for the snapshot type. For snapshot lists use the value 0 for “decrement to previous snapshot in list” and the value 127 for “increment to next snapshot in list”.
14 [W]	Inactive state snapshot ID or snapshot list index (0 – N). Must be a valid value for the snapshot type. For snapshot lists use the value 0 for “decrement to previous snapshot in list” and the value 127 for “increment to next snapshot in list”.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x01, 0x03                     // command flags and ID
0x00, 0x0E                     // data length (14)
0x01                           // command version number (1)
0x01                           // hardware interface type: footswitch (1)
0x04                           // footswitch ID (4)
0x02                           // jack ID (2)
0x01                           // jack type: phone jack (1)
0x02                           // location on jack: ring (2)
0x47                           // flags: enabled, input, supports input and output
```

```
0x00, 0x05                  // debounce time (5 ms)
0x01                        // function code: snapshot selection (1)
0x7F                        // snapshot type: scene (127)
0x03                        // flags: active and inactive state both use snapshot list
0x01                        // active state snapshot list index (1)
0x02                        // inactive state snapshot list index (2)
0xxx                        // checksum
0xF7                        // footer
```

For function code = 2 (mute group selection):

Footswitch Mute Group Selection block:

Byte	Description
11 [W]	Active state mute group ID (1 – N). Must be a valid mute group ID for the device. Use 0 to deactivate all mute groups.
12 [W]	Inactive state mute group ID (1 – N). Must be a valid mute group ID for the device. Use 0 to deactivate all mute groups.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E  // header
0x00, 0x05                    // product ID
0x01, 0x02, 0x03, 0x04, 0x05  // serial number
0x00, 0x00                    // transaction ID
0x01, 0x03                    // command flags and ID
0x00, 0x0C                    // data length (12)
0x01                          // command version number (1)
0x01                          // hardware interface type: footswitch (1)
0x03                          // footswitch ID (3)
0x01                          // jack ID (1)
0x01                          // jack type: phone jack (1)
0x01                          // location on jack: tip (1)
0x65                          // flags: enabled, momentary, input, supports input only
0x00, 0x0A                    // debounce time (10 ms)
0x02                          // function code: mute group selection (2)
0x01                          // active state mute group ID (1)
0x00                          // inactive state mute group ID (0)
0xxx                          // checksum
0xF7                          // footer
```

For function code = 3 (automatic failover setting):

Footswitch Automatic Failover Setting block:

Byte	Description
11 [W]	Active state setting: 0: no effect 1: arm automatic failover system 2: disarm automatic failover system
12 [W]	Inactive state setting: 0: no effect 1: arm automatic failover system 2: disarm automatic failover system

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x01, 0x03                      // command flags and ID
0x00, 0x0C                      // data length (12)
0x01                            // command version number (1)
0x01                            // hardware interface type: footswitch (1)
0x03                            // footswitch ID (3)
0x01                            // jack ID (1)
0x01                            // jack type: phone jack (1)
0x01                            // location on jack: tip (1)
0x65                            // flags: enabled, momentary, input, supports input only
0x00, 0x0A                      // debounce time (10 ms)
0x03                            // function code: automatic failover setting (3)
0x01                            // active state arms failover system (1)
0x02                            // inactive state disarms failover system (2)
0xxx                            // checksum
0xF7                            // footer
```

For hardware interface type = 2 (mute group):

Mute groups are used to silence audio outputs without changing the volume or the mute control for individual audio channels. Each mute group has a bitmask for each audio port indicating which output channels should be muted when that mute group is activated. When deactivated, all output channels return to the state they were in before the mute group was activated.

Byte #3: mute group ID (1 - N). Must be a valid mute group ID for this device.
Byte #4: number of mute group blocks that follow (1 – N).
Bytes #5 – N: mute group blocks.

Each mute group block begins with a common header:

Byte	Description
1	Size of this block (including this byte).
2	Mute group block type: 1: audio port
3 - N	Dependent on mute group block type (see tables below).

For mute group block type = 1 (audio port):

Byte	Description
3 - 4	audio port ID (1 – N).
5	Number of bytes in bitmap that follows.
6 - N	Bitmap indicating mute status for audio output channels on this audio port. A bit is set if the channel should be muted when this mute group is activated. A bit is clear if the channel should be left untouched when the mute group is activated. Bit 0 is channel #1, bit 1 is channel #2, etc. Least significant byte is first, most significant byte is last. The most significant 4 bits of each byte must be 0 which allows 4 channels to be specified per byte. Unused bits should be set to 0. The number of channels is given in the RetAudioPortParm command. The number of bytes (in the bitmap) is an even value so that the bitmap (when unpacked) is a multiple of 8 bits: number of bytes (using INTEGER math) = (((number of output channels - 1) / 8) + 1) x 2

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x01, 0x03                      // command flags and ID
0x00, 0x12                      // data length (18)
0x01                            // command version number (1)
0x02                            // hardware interface type: mute group (2)
0x01                            // mute group ID (1)
0x02                            // number of mute group blocks that follow (2)
0x07                            // block #1: size of this block (7)
0x01                            // block #1: block type = audio port (1)
0x00, 0x01                      // block #1: audio port ID (1)
0x02                            // block #1: number of bytes in bitmap that follows (2)
0x0F                            // block #1: mute flags for output channels 4 – 1
0x05                            // block #1: mute flags for output channels 8 – 5
0x07                            // block #2: size of this block (7)
0x01                            // block #2: block type = audio port (1)
```

0x00, 0x02 // block #2: audio port ID (2)
0x02 // block #2: number of bytes in bitmap that follows (2)
0x03 // block #2: mute flags for output channels 4 – 1
0x0C // block #2: mute flags for output channels 8 – 5
0xxx // checksum
0xF7 // footer

For hardware interface type = 3 (automatic failover):

Automatic failover is used to switch audio and MIDI data from a main system to a backup system automatically should the main system fail for some reason. Typically this would be configured such that the main (or primary) system provides audio and MIDI data on USB device port #1 while the backup (or secondary) system provides an identical synchronized copy of the same audio and MIDI data on USB device port #2. Scenes are used to configure the audio and MIDI routing for the main and backup systems. The automatic failover system has a number of operating modes and parameters to control switching between scenes and detecting when the main system has failed. The system can also be configured to automatically arm itself when the main system is stable.

Byte #3: always 0.
Bytes #4 – N: automatic failover block.

Automatic Failover block:

Byte	Description
4 – 5	Main (primary) audio port ID (1 – N). Must be a USB device port. Cannot be the same as Backup audio port.
6 – 7	Backup (secondary) audio port ID (1 – N). Must be a USB device port. Cannot be the same as Main audio port.
8	Snapshot type (1 – N). Must be a supported snapshot type for the device.
9	Flags: bits 7 - 5: reserved (always 0) bit 4: set this flag to automatically generate snapshots for the automatic failover system from the current settings, this flag is write-only, it will always read back as clear bit 3: set if system should automatically select the main snapshot ID when automatically arming itself (only used if bit 2 is also set) bit 2: set if the system should automatically arm itself when the main system is stable bit 1: set if Backup ID (byte 11) refers to snapshot list index, clear if Backup ID refers to snapshot ID. bit 0: set if Main ID (byte 10) refers to snapshot list index, clear if Main ID refers to snapshot ID.
10	Main snapshot ID or snapshot list index (0 – N). Must be a valid value for the snapshot type. For snapshot lists use the value 0 for “decrement to previous snapshot in list” and the value 127 for “increment to next snapshot in list”.
11	Backup snapshot ID or snapshot list index (0 – N). Must be a valid value for the snapshot type. For snapshot lists use the value 0 for “decrement to previous snapshot in list” and the

Byte	Description
	value 127 for “increment to next snapshot in list”.
12	Trigger mode for switching from main to backup system: 1: audio only 2: MIDI only 3: audio or MIDI 4: audio and MIDI
13	Audio trigger level for switching from main to backup system. If the host fails to meet the data requirements then the automatic failover system will switch to the backup system. 1: host has enumerated the device and is sending USB start of frame (SOF) 2: host is sending any data (could be digital silence) to USB audio OUT endpoint 3: host is sending non-zero data to a specific audio channel (see byte 14)
14	Audio trigger channel for main system (1 – N). Must be a valid audio channel number. Only used if audio trigger level = 3 (see byte 13).
15-16	Audio timeout value (1 – 16363). Number of USB frames that must fail to meet the audio trigger level before switching from main to backup system.
17	MIDI trigger level for switching from main to backup system. If the host fails to meet the data requirements then the automatic failover system will switch to the backup system. 1: host has enumerated the device and is sending USB start of frame (SOF) 2: host is sending any data to USB MIDI OUT endpoint 3: host is sending data to a specific MIDI port (see byte 18)
18	MIDI trigger port for main system (1 – 16). Must be a valid MIDI port number for the USB port. Only used if MIDI trigger level = 3 (see byte 17).
19-20	MIDI timeout value (1 – 16383). Number of USB frames that must fail to meet the MIDI trigger level before switching from main to backup system.
21	MIDI recovery mode (0 – N). Defines the method to be used to deal with lost MIDI messages when switching from main to backup system. 0: disable MIDI recovery mode 1: send specific MIDI messages to selected ports and channels

Bytes 22 – N: depends on MIDI recovery mode.

For mode 0: no additional bytes are sent.

For mode 1: this mode sends specific MIDI messages to selected ports and channels when the automatic failover system switches from the main to the backup system. The intent is to silence any outstanding “note on” messages received from the main system before switching over to the backup system. It is generally sufficient to send either “all sound off + sustain pedal off” or “all notes off + sustain pedal off” on all MIDI channels being used.

MIDI recovery mode 1 block:

Byte	Description
1	Message flags. Defines the MIDI messages to be sent. bits 7 - 4: reserved (always 0) bit 3: send reset all controllers message (0xBn 0x79 0x00) bit 2: send all notes off message (0xBn 0x7B 0x00) bit 1: send all sound off message (0xBn 0x78 0x00) bit 0: send sustain pedal off message (0xBn 0x40 0x00)
2 – 4	Bitmask indicating which MIDI channels to use for sending messages (0x0001 = MIDI channel 1, 0x0002 = MIDI channel 2 … 0x8000 = MIDI channel 16). More than 1 bit can be set (16-bit value encoded in 3 bytes).
5	Number of bytes in bitmask field that follows.
6 – N	Bitmap used to define which MIDI ports will have messages sent to them. Set the appropriate bit to send MIDI messages to that port. Clear the appropriate bit to not send any MIDI messages to that port. This bitmap is identical in format to that used for MIDI port routing (see RetMIDIPortRoute command). Bit 0 is port #1, bit 1 is port #2, etc. Least significant byte is first, most significant byte is last. The most significant 4 bits of each byte must be 0 which allows 4 ports to be specified per byte. Unused bits should be set to 0. The number of ports is given in the RetMIDIInfo command. The number of bytes (in the bitmap) is an even value so that the bitmap (when unpacked) is a multiple of 8 bits: number of bytes (using INTEGER math) = (((number of ports - 1) / 8) + 1) x 2

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x01, 0x03                      // command flags and ID
0x00, 0x1C                      // data length (28)
0x01                            // command version number (1)
0x03                            // hardware interface type: automatic failover (3)
0x00                            // always 0
0x00, 0x01                      // main port ID (1)
0x00, 0x02                      // backup port ID (2)
0x7F                            // snapshot type (scene)
0x04                            // flags: use snapshot IDs, enable auto-arm
0x01                            // main snapshot ID (1)
0x02                            // backup snapshot ID (2)
0x04                            // trigger mode: audio and MIDI (4)
0x03                            // audio trigger mode: use specific audio channel (3)
0x0A                            // use audio channel #10
0x00, 0x04                      // audio timeout value: 4 frames
0x02                            // MIDI trigger mode: use any data (2)
0x01                            // MIDI port (1)
0x00, 0x64                      // MIDI timeout value: 100 frames
0x01                            // MIDI recovery mode (1)
```

```
0x03                    // mode 1 flags: send all sound off + sustain pedal off messages
0x03, 0x7F, 0x7F        // mode 1 MIDI channels: send messages to all 16 MIDI channels
0x02                    // mode 1 bitmap size that follows (2)
0x00, 0x01              // mode 1 MIDI port bitmap: send messages to port 1 only
0xxx                    // checksum
0xF7                    // footer
```

For hardware interface type = 4 (tone generator):

Tone generator is used to send a sine wave test signal to specific audio outputs. The frequency of the test tone, the amplitude, and the outputs to be used may be specified.

Byte #3: always 0.
Bytes #4 – N: tone generator block.

Tone Generator block:

Byte	Description
4 – 6	Minimum frequency (Hz) supported by tone generator (16 bit value encoded in 3 bytes).
7 – 9	Maximum frequency (Hz) supported by tone generator (16 bit value encoded in 3 bytes).
10 – 12	Frequency resolution (Hz) supported by tone generator (16 bit value encoded in 3 bytes). For example, a value of 10 indicates that frequencies must be in multiples of 10 Hz (e.g. 100 Hz, 110 Hz, 120 Hz, etc.)
13 – 15	Minimum volume (dB) supported by tone generator (16 bit value encoded in 3 bytes).
16 – 18	Maximum volume (dB) supported by tone generator (16 bit value encoded in 3 bytes).
19 – 21	Volume resolution (dB) supported by tone generator (16 bit value encoded in 3 bytes). For example, a value of 0.5 indicates that attenuation must be in multiples of 0.5 dB (e.g. -3.0 dB, -3.5 dB, -4.0 dB, etc.)

The volume values are 16 bit 2's complement numbers (8.8 format) that can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001). Resolution can only have positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x01, 0x03                      // command flags and ID
0x00, 0x15                      // data length (21)
0x01                            // command version number (1)
0x04                            // hardware interface type: tone generator (4)
0x00                            // always 0
0x00, 0x00, 0x14                // minimum frequency (20 Hz)
```

```
0x00, 0x4E, 0x10        // maximum frequency (10 kHz)
0x00, 0x00, 0x0A        // frequency resolution (10 Hz)
0x02, 0x40, 0x00        // minimum volume (-96 dB)
0x00, 0x14, 0x00        // maximum volume (+6 dB)
0x00, 0x00, 0x40        // volume resolution (0.25 dB)
0xxx                    // checksum
0xF7                    // footer
```

9.5. GetHardwareValue (Command ID = 0x84)

This command is used to get the current value from a hardware interface element. Devices that support this command will respond with a RetHardwareValue message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x84.
Data length is 2.

Byte #1: hardware interface type (1 - N).

For hardware interface type = 1 (footswitch):

Byte #2: always 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x41, 0x04                      // command flags and ID
0x00, 0x02                      // data length
0x01                            // hardware interface type: footswitch (1)
0x00                            // always 0
0xxx                            // checksum
0xF7                            // footer
```

For hardware interface type = 2 (mute group):

Byte #2: always 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x41, 0x04                      // command flags and ID
0x00, 0x02                      // data length
0x02                            // hardware interface type: mute group (2)
```

```
0x00                        // always 0
0xxx                        // checksum
0xF7                        // footer
```

For hardware interface type = 3 (automatic failover):

Byte #2: always 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x41, 0x04                     // command flags and ID
0x00, 0x02                     // data length
0x03                           // hardware interface type: automatic failover (3)
0x00                           // always 0
0xxx                           // checksum
0xF7                           // footer
```

For hardware interface type = 4 (tone generator):

Byte #2: always 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x41, 0x04                     // command flags and ID
0x00, 0x02                     // data length
0x04                           // hardware interface type: tone generator (4)
0x00                           // always 0
0xxx                           // checksum
0xF7                           // footer
```

9.6. RetHardwareValue / SetHardwareValue (Command ID = 0x85)

RetHardwareValue is sent by devices in response to a GetHardwareValue command.

SetHardwareValue is sent by a host to set the values for a specific hardware interface element. Not all hardware interface types support this command.

Command Data

Command flags are Answer for RetHardwareValue, Write for SetHardwareValue.
Command ID is 0x85.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2: hardware type (1 – N).

For hardware interface type = 1 (footswitch):

RetHardwareValue returns the state of all footswitches in a single command. SetHardwareValue is not supported for footswitches.

Byte #3: always 0.
Byte #4: number of bytes in bitmap that follows.
Bytes #5 – N: bitmap indicating active state of each footswitch (active = TRUE, inactive = FALSE). Bit 0 is footswitch #1, bit 1 is footswitch #2, etc. Least significant byte is first, most significant byte is last. The most significant 4 bits of each byte must be 0 which allows 4 footswitches to be specified per byte. Unused bits should be set to 0. The number of footswitches is given in the RetHardwareGlobalParm command. The number of bytes (in the bitmap) is an even value so that the bitmap (when unpacked) is a multiple of 8 bits:

number of bytes (using INTEGER math) = (((number of footswitches - 1) / 8) + 1) x 2

Example (device has 4 footswitches):

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x01, 0x05                     // command flags and ID
0x00, 0x06                     // data length (6)
0x01                           // command version number (1)
0x01                           // hardware interface type: footswitch (1)
0x00                           // always 0
0x02                           // number of bytes in bitmap that follows (2)
0x05                           // state of footswitches 4 through 1 (1 and 3 are active)
0x00                           // state of footswitches 8 through 5 (padding)
0xxx                           // checksum
0xF7                           // footer
```

For hardware interface type = 2 (mute group):

RetHardwareValue returns the currently active mute group ID (1 – N) or 0 if no mute group is active. SetHardwareValue is used to activate a mute group ID (1 – N) or deactivate all mute groups (0).

Byte #3: always 0.
Byte #4: active mute group ID (1 – N) or 0 if no mute group is active.

Example (mute group #1 is active):

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
```

```
0x00, 0x00          // transaction ID
0x01, 0x05          // command flags and ID
0x00, 0x04          // data length (4)
0x01                // command version number (1)
0x02                // hardware interface type: mute group (2)
0x00                // always 0
0x01                // mute group ID (1)
0xxx                // checksum
0xF7                // footer
```

For hardware interface type = 3 (automatic failover):

RetHardwareValue returns the current state of the ARM and ALARM flags. The ARM flag is set if the automatic failover system is armed. The ALARM flag is set if the automatic failover system has detected a failure in the main system and has switched over to the backup system.

SetHardwareValue is used to set or clear the ARM flag (to arm the automatic failover system or to disable it respectively) and to clear the ALARM flag. It is not possible to set the ALARM flag with this command; the ALARM flag can only be set by the automatic failover system. Setting the ARM flag will automatically clear the ALARM flag. Only the first 5 bytes need to be sent since the remaining bytes are read-only status indicators.

Byte #3: always 0.
Byte #4: value flags:
  bits 7 – 2 = reserved (always zero)
  bit 1 = set if ALARM flag is included in byte #5
  bit 0 = set if ARM flag is included in byte #5

Byte #5: status of each of the flags indicated in byte #4:

Bit	Description
7 - 2	reserved (always zero)
1 [W]	ALARM: set if automatic failover system has detected a failure in the main system and has switched over to the backup system. Write 0 to clear the ALARM flag. This flag cannot be set by writing 1.
0 [W]	ARM: set if the automatic failover system is armed. Clear this flag to disable the automatic failover system. Setting this flag will arm the system and automatically clear the ALARM flag and select the Main snapshot ID or snapshot list index.

Byte #6: current status of main audio (see the table below).
Byte #7: current status of main MIDI (see the table below).
Byte #8: current status of backup audio (see the table below).
Byte #9: current status of backup MIDI (see the table below).

Value	Description
0	Host is not connected or is not sending any USB data.
1	Host has enumerated the device and is sending USB start of frame (SOF).
2	Host is sending data to the USB audio or MIDI OUT endpoint.
3	Host is sending non-zero data to the specified audio channel (main audio only) or sending data to the specified MIDI port (main MIDI only).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x01, 0x05                      // command flags and ID
0x00, 0x09                      // data length (9)
0x01                            // command version number (1)
0x03                            // hardware interface type: automatic failover (3)
0x00                            // always 0
0x03                            // value flags: alarm and arm flags are in next byte
0x01                            // status flags: system is armed (alarm is not set)
0x03                            // host is sending non-zero data to the specified audio channel (main)
0x02                            // host is sending data to MIDI OUT endpoint (main)
0x02                            // host is sending data to audio OUT endpoint (backup)
0x01                            // host is sending start of frame, no MIDI data is being sent (backup)
0xxx                            // checksum
0xF7                            // footer
```

For hardware interface type = 4 (tone generator):

RetHardwareValue returns the current frequency and volume of the tone generator and bitmaps indicating which audio channels are outputting the tone. Bitmaps for all audio ports are always returned.

SetHardwareValue is used to set the frequency and volume of the tone generator and to select which audio channels are outputting the tone.

Byte #3: always 0.

Byte #4: value flags:
  bits 7 – 2 = reserved (always zero)
  bit 1 = set if tone generator amplitude value is included (bytes 8 – 10)
  bit 0 = set if tone generator frequency value is included (bytes 5 – 7)

Bytes #5 - 7: tone generator frequency (Hz),16 bit value encoded in 3 bytes.
Bytes #8 - 10: tone generator amplitude (dB),16 bit 2's complement value (8.8 format) encoded in 3 bytes.
Byte #11: number of tone generator output blocks that follow (0 – N).

Tone Generator Output block:

Byte	Description
1 – 2	audio port ID (1 – N)
3	number of bytes in the bitmap that follows
4 – N	Bitmap indicating which output channels are connected to the tone generator. Bit 0 is output channel #1, bit 1 is output channel #2, etc. Least significant byte is first, most significant byte is last. The most significant 4 bits of each byte must be 0 which allows 4 channels to be specified per byte. Unused bits should be set to 0. The number of output channels is given in the RetAudioPortParm command. The number of bytes (in the bitmap) is an even value so that the bitmap (when unpacked) is a multiple of 8 bits: number of bytes (using INTEGER math) = (((number of output channels - 1) / 8) + 1) x 2

Example (audio port #3 has 12 outputs, tone generator is connected to outputs 1, 2, 11, 12:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x01, 0x05                     // command flags and ID
0x00, 0x12                     // data length (18)
0x01                           // command version number (1)
0x04                           // hardware interface type: tone generator (4)
0x00                           // always 0
0x03                           // tone generator frequency and volume values are included
0x00, 0x03, 0x38               // tone generator frequency (440 Hz)
0x03, 0x7A, 0x00               // tone generator volume (-3 dB = 0xFD00 in 8.8 format)
0x01                           // number of tone generator output blocks that follow (1)
0x00, 0x03                     // audio port ID (3)
0x04                           // number of bytes in the bitmap that follows (4)
0x03                           // bitmap for channels 4 – 1 (channels 1 and 2 are connected)
0x00                           // bitmap for channels 8 – 5
0x0C                           // bitmap for channels 12 – 9 (channels 11 and 12 are connected)
0x00                           // bitmap for channels 16 – 13 (padding)
0xxx                           // checksum
0xF7                           // footer
```

10. Audio Commands V1 [deprecated]

Support for these commands will be removed from all products in the near future. These commands were used in early firmware versions of iConnectMIDI2+ and iConnectMIDI4+.

The following commands are defined for protocol version = 1.

10.1. GetAudioInfo (Command ID = 0x30)

This command is used to query a device about audio parameters. Devices that support this command will respond with a RetAudioInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x30.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x30                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

10.2. RetAudioInfo (Command ID = 0x31)

RetAudioInfo is sent by devices in response to a GetAudioInfo command. It contains basic information that a host will need to use to communicate with this device for all other audio related messages. A host should cache this information and use it for all further communication with this device.

Command Data

Command flags are Answer.
Command ID is 0x31.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: number of audio ports supported by this device (0 - N).
Byte# 4: number of audio capable USB device jacks supported by this device (0 - N).
Byte# 5: number of audio capable USB host jacks supported by this device (0 - N).

Byte# 6: number of audio capable ethernet jacks supported by this device (0 - N).
Byte# 7: number of audio ports supported by each USB host jack (0 - N).
Byte# 8: number of audio ports supported by each ethernet jack (0 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x31                      // command flags and ID
0x00, 0x08                      // data length (8)
0x01                            // command version number (1)
0x00, 0x09                      // device has 9 audio capable ports
0x03                            // device has 3 audio capable USB device jacks
0x01                            // device has 1 audio capable USB host jack
0x01                            // device has 1 audio capable ethernet jack
0x02                            // device supports 2 audio ports on each USB host jack
0x04                            // device supports 4 audio ports on each ethernet jack
0xxx                            // checksum
0xF7                            // footer
```

10.3. GetAudioCfgInfo (Command ID = 0x32)

This command is used to query a device about audio configurations. Devices that support this command will respond with a RetAudioCfgInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x32.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x32                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

10.4. RetAudioCfgInfo / SetAudioCfgInfo (Command ID = 0x33)

RetAudioCfgInfo is sent by devices in response to a GetAudioCfgInfo command. It contains information that a host will need to use to communicate with this device for other audio related messages. A host should cache this information and use it for further communication with this device.

SetAudioCfgInfo is sent by a host to a device to set the current configuration. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored. Parameters in RED underline require the device to be reset before taking effect.

Command Data

Command flags are Answer for RetAudioCfgInfo, Write for SetAudioCfgInfo.
Command ID is 0x33.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The host only needs to send up to (and including) byte #8, all bytes beyond #8 are read-only and are ignored.

Byte #2: minimum number of audio frames that can be buffered (1 - N).
Byte #3: maximum number of audio frames that can be buffered (1 - N).
Byte #4 [W]: current number of audio frames to buffer (1 - N).
Byte #5: minimum allowed value for sync factor (1 - N).
Byte #6: maximum allowed value for sync factor (1 - N).
Byte #7 [W]: current sync factor value (1 - N).
Byte #8 [W]: number of the currently active configuration (1 - N).
Byte #9: number of configuration blocks that follow (0 - N).

Bytes #10-N: configurations blocks. Bytes 10 and up are used only if byte #9 is not zero. Four bytes are used for each configuration block as follows:

Byte	Description
1	configuration number (1 - N)
2	bit depth code: 1 = 4 bit, 2 = 8 bit, 3 = 12 bit, 4 = 16 bit bit depth code: 5 = 20 bit, 6 = 24 bit, 7 = 28 bit, 8 = 32 bit
3	sample rate code: 1 = 11025, 3 = 22050, 5 = 44100, 7 = 88200 sample rate code: 2 = 12000, 4 = 24000, 6 = 48000, 8 = 96000
4	number of audio channels (0 - N)

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x33                      // command flags and ID
0x00, 0x29                      // data length (41)
```

```
0x01                        // command version number (1)
0x01, 0x04, 0x02            // minimum, maximum, and current # of audio frames to buffer
0x01, 0x04, 0x02            // minimum, maximum, and current sync factor
0x02                        // currently active configuration is #2
0x08                        // number of configuration blocks that follow (8)
0x01, 0x04, 0x05, 0x04      // configuration block #1: 16 bit, 44100, 4 channels
0x02, 0x04, 0x05, 0x08      // configuration block #2: 16 bit, 44100, 8 channels
0x03, 0x04, 0x06, 0x04      // configuration block #3: 16 bit, 48000, 4 channels
0x04, 0x04, 0x06, 0x08      // configuration block #4: 16 bit, 48000, 8 channels
0x05, 0x04, 0x06, 0x04      // configuration block #5: 16 bit, 88200, 4 channels
0x06, 0x04, 0x08, 0x04      // configuration block #6: 16 bit, 96000, 4 channels
0x07, 0x06, 0x05, 0x04      // configuration block #7: 24 bit, 44100, 4 channels
0x08, 0x06, 0x06, 0x04      // configuration block #8: 24 bit, 48000, 4 channels
0xxx                        // checksum
0xF7                        // footer
```

10.5. GetAudioPortInfo (Command ID = 0x34)

This command is used to query a device about a specific audio port. Devices that support this command will respond with a RetAudioPortInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x34.
Data length is 2.

Bytes #1-2: audio port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x34                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // audio port ID
0xxx                            // checksum
0xF7                            // footer
```

10.6. RetAudioPortInfo / SetAudioPortInfo (Command ID = 0x35)

RetAudioPortInfo is sent by devices in response to a GetAudioPortInfo command.

SetAudioPortInfo is sent by a host to a device to set the values of some of the port parameters. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored. Parameters in RED underline require the device to be reset before taking effect.

Command Data

Command flags are Answer for RetAudioPortInfo, Write for SetAudioPortInfo.
Command ID is 0x35.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

For version = 1, minimum length is 13, actual length depends on the port name (a 7-bit ASCII string, not NULL terminated). Use the data length field to determine the string length.

Bytes #2-3: audio port ID (1 - N).

Byte #4: port type:

Value	Description
2	USB device
3	USB host
4	ethernet

Bytes #5-8: port info, depends on port type (byte #4):

Type	Description
USB device	Byte #5: USB device jack # (1 - N) Byte #6-8: reserved (always 0)
USB host	Byte #5: USB host jack # (1 - N) Byte #6: host port # (1 - N) Byte #7-8: reserved (always 0)
ethernet	Byte #5: ethernet jack # (1 - N) Byte #6: port # (1 - N) Byte #7-8: reserved (always 0)

Byte #9: maximum length allowed for port name, 0 if read-only (0 - 127).

Bytes #10-13: port specific options, format of these bytes depends on port type (byte #4). For USB device ports the four bytes represent a bitmap as follows

Bit	Description
31 - 4	reserved (always zero)
3 [W]	set if port is currently enabled for audio with iOS devices
2 [W]	set if port is currently enabled for audio with PC/Mac
1	set if port supports audio with iOS devices
0	set if port supports audio with PC/Mac

Bytes #14-N [W]: port name, 7-bit ASCII string, not NULL terminated. Name must follow the same rules as used for device name (see RetInfo command).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x35                      // command flags and ID
0x00, 0x11                      // data length (17)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID
0x02                            // port type = USB device
0x01                            // port is first USB device jack
0x00, 0x00, 0x00                // reserved for USB device port
0x04                            // maximum length allowed for port name (4 ASCII characters)
0x00, 0x00, 0x00, 0x07          // port supports PC and iOS audio but iOS audio is currently disabled
0x55, 0x53, 0x42, 0x31          // port name, the ASCII string “USB1”
0xxx                            // checksum
0xF7                            // footer
```

10.7. GetAudioPortCfgInfo (Command ID = 0x36)

This command is used to query a device about audio port configurations. Devices that support this command will respond with a RetAudioPortCfgInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x36.
Data length is 2.

Bytes #1-2: audio port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x36                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // audio port ID
0xxx                            // checksum
0xF7                            // footer
```

10.8. RetAudioPortCfgInfo / SetAudioPortCfgInfo (Command ID = 0x37)

RetAudioPortCfgInfo is sent by devices in response to a GetAudioPortCfgInfo command. It contains information related to the number of audio channels that this port supports for each of the different audio configurations (from RetAudioCfgInfo command).

SetAudioPortCfgInfo is sent by a host to a device to set the current port configuration. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored. Parameters in RED underline require the device to be reset before taking effect.

Command Data

Command flags are Answer for RetAudioPortCfgInfo, Write for SetAudioPortCfgInfo.
Command ID is 0x37.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The host only needs to send up to (and including) byte #5, all bytes beyond #5 are read-only and are ignored.

Bytes #2-3: audio port ID (1 - N).

Byte #4 [W]: number of destination (output) channels to use for this port (0 - N). Must be a legal value for the current audio configuration.
Byte #5 [W]: number of source (input) channels to use for this port (0 - N). Must be a legal value for the current audio configuration.
Byte #6: number of port configuration blocks that follow (0 - N).

Bytes #7-N: port configurations blocks. Bytes 7 and up are used only if byte #6 is not zero. There is one port configuration block for each audio configuration (from RetAudioCfgInfo command). Five bytes are used for each port configuration block as follows:

Byte	Description
1	audio configuration number (1 - N, from RetAudioCfgInfo command)

Byte	Description
2	minimum number of destination (output) channels (0 - N)
3	maximum number of destination (output) channels (0 - N)
4	minimum number of source (input) channels (0 - N)
5	maximum number of source (input) channels (0 - N)

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x37                      // command flags and ID
0x00, 0x2E                      // data length (46)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID
0x04                            // number of destination (output) channels to use for this port (4)
0x04                            // number of source (input) channels to use for this port (4)
0x08                            // number of port configuration blocks that follow (8)
0x01, 0x01, 0x04, 0x00, 0x04    // port configuration block #1 (16/44/4): 1/4 out, 0/4 in
0x02, 0x01, 0x08, 0x00, 0x08    // port configuration block #2 (16/44/8): 1/8 out, 0/8 in
0x03, 0x01, 0x04, 0x00, 0x04    // port configuration block #3 (16/48/4): 1/4 out, 0/4 in
0x04, 0x01, 0x08, 0x00, 0x08    // port configuration block #4 (16/48/8): 1/8 out, 0/8 in
0x05, 0x01, 0x02, 0x00, 0x02    // port configuration block #5 (16/88/4): 1/2 out, 0/2 in
0x06, 0x01, 0x02, 0x00, 0x02    // port configuration block #6 (16/96/4): 1/2 out, 0/2 in
0x07, 0x01, 0x04, 0x00, 0x04    // port configuration block #7 (24/44/4): 1/4 out, 0/4 in
0x08, 0x01, 0x04, 0x00, 0x04    // port configuration block #8 (24/48/2): 1/4 out, 0/4 in
0xxx                            // checksum
0xF7                            // footer
```

10.9. GetAudioPortPatchbay (Command ID = 0x38)

This command is used to query a device about audio port patchbay setup. Devices that support this command will respond with a RetAudioPortPatchbay message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x38.
Data length is 2.

Bytes #1-2: audio port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x38                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // audio port ID
0xxx                            // checksum
0xF7                            // footer
```

10.10. RetAudioPortPatchbay / SetAudioPortPatchbay (Command ID = 0x39)

RetAudioPortPatchbay is sent by devices in response to a GetAudioPortPatchbay command. It contains the patchbay setup for a specific audio port.

SetAudioPortCfgInfo is sent by a host to a device to configure the patchbay for a specific audio port.

Command Data

Command flags are Answer for RetAudioPortPatchbay, Write for SetAudioPortPatchbay.
Command ID is 0x39.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).

Byte #4: number of patchbay configuration blocks that follow (0 - N).

Bytes #5-N: patchbay configurations blocks. Bytes 5 and up are used only if byte #4 is not zero. If byte #4 is not zero, there should be one patchbay configuration block for each destination (output) channel for this port. Four bytes are used for each port configuration block as follows:

Byte	Description
1	destination (output) channel number (1 - N)
2	source (input) channel number (1 - N)
3 - 4	source (input) port ID (1 - N)

All values in each patch configuration block must be valid for the source port ID, source channel number, and destination channel number. Use 0x00 for bytes #2-4 to indicate “no connection” (i.e. no source (input) is connected to the destination (output) channel).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x00, 0x39                     // command flags and ID
0x00, 0x14                     // data length (20)
0x01                           // command version number (1)
0x00, 0x01                     // audio port ID
0x04                           // number of patchbay configuration blocks that follow (4)
0x01, 0x03, 0x00, 0x02         // block #1: output #1 is connected to port #2 channel 3
0x02, 0x01, 0x00, 0x03         // block #2: output #2 is connected to port #3 channel 1
0x03, 0x02, 0x00, 0x03         // block #3: output #3 is connected to port #3 channel 2
0x04, 0x00, 0x00, 0x00         // block #4: output #4 is not connected to anything
0xxx                           // checksum
0xF7                           // footer
```

10.11. GetAudioClockInfo (Command ID = 0x3A)

This command is used to query a device about audio clock sources. Devices that support this command will respond with a RetAudioClockInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x3A.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x40, 0x3A                     // command flags and ID
0x00, 0x00                     // data length
0xxx                           // checksum
0xF7                           // footer
```

10.12. RetAudioClockInfo / SetAudioClockInfo (Command ID = 0x3B)

RetAudioClockInfo is sent by devices in response to a GetAudioClockInfo command. It contains information regarding audio clock sources for the device.

SetAudioClockInfo is sent by a host to a device to set the audio clock source. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored. The host only needs to send up to (and including) byte #2, all bytes beyond #2 are read-only and are ignored. Parameters in RED underline require the device to be reset before taking effect.

Command Data

Command flags are Answer for RetAudioClockInfo, Write for SetAudioClockInfo.
Command ID is 0x3B.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2 [W]: number of the active audio clock source block (1 - N).
Byte #3: number of audio clock source blocks that follow (0 - N).
Bytes #4-N: audio clock source blocks. Bytes 4 and up are used only if byte #3 is not zero. Four bytes are used for each audio clock source block as follows:

Byte	Description
1	audio clock source number (1 - N)
2	clock source type: 1 = internal clock 2 = audio port clock
3 - 4	For clock source type = internal clock: Byte 3: internal clock ID (1 - N) Byte 4: reserved (always 0) For clock source type = audio port clock: Byte 3: audio port ID (1 - N) Byte 4: reserved (always 0)

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x3B                      // command flags and ID
0x00, 0x0F                      // data length (15)
0x01                            // command version number (1)
0x02                            // active audio clock source block is #2
0x03                            // number of audio clock blocks that follow (3)
0x01, 0x01, 0x01, 0x00          // audio clock source block #1: internal clock #1
0x02, 0x02, 0x01, 0x00          // audio clock source block #2: USB audio port #1
0x03, 0x02, 0x02, 0x00          // audio clock source block #3: USB audio port #2
```


0xxx // checksum
0xF7 // footer

11. Product IDs

PID	Device
1 (0x0001)	iConnectMIDI (does not support the commands in this document)
2 (0x0002)	mio10
3 (0x0003)	mio
4 (0x0004)	iConnectMIDI1
5 (0x0005)	iConnectMIDI2+
6 (0x0006)	iConnectMIDI4+
7 (0x0007)	iConnectAUDIO4+
8 (0x0008)	iConnectAUDIO2+
9 (0x0009)	mio2
10 (0x000A)	mio4
11 (0x000B)	PlayAUDIO12
12 (0x000C)	
13 (0x000D)	ConnectAUDIO2/4

12. History

#	Date/Author	Changes
23	17-11-01/ S. Juskiw	1. Added GetUserData commands. 2. Added Tone Generator to Hardware Interface Commands. 3. Added flags to allow Automatic Failover system to automatically arm itself. 4. Added GetRTPMIDIConnectionParm command. 5. Added flag to SetHardwareParm for Automatic Failover system to automatically create failover compatible snapshots.
22	17-07-20/ S. Juskiw	1. Fixed a few typos in the Hardware Interface Commands. 2. Added Automatic Failover to Hardware Interface Commands. 3. GetDevice command no longer does network discovery. A new command for network discovery will be added later.
21	17-04-06/ S. Juskiw	1. CreateSnapshot now supports scenes. 2. Added mixer controls to snapshots. 3. Removed “audio control snapshot block type” from audio control snapshots.
20	17-01-13/ S. Juskiw	1. Added rules for device name, MIDI/audio port name, audio channel name and many other object names to address issues discovered with various operating systems that don’t like short names or names with special characters in them. See RetInfo command for details. 2. Added some clarifications to the sections on MIDI filter and remap messages. 3. Added Get/Ret/SetDeviceMode commands. 4. Fixed a number of errors in the Audio Commands V2 section (inputs and outputs were often swapped). The nomenclature for signal flow has been enhanced from “inputs” and “outputs” to “sources/inputs” and “destinations/outputs” respectively. Added some notes to explain how sources and destinations are defined in RetAudioPortParm message. 5. Updated Audio Commands V1 to match the “sources/inputs” and “destinations/outputs” nomenclature used in Audio Commands V2.
19	16-11-24/ S. Juskiw	1. Added command version #2 to RetMixerPortParm. 2. Fixed an error in RetAutomationControlDetail message (MIDI channel number is 1-16 not 0-15). 3. Added Operator Order to Operator Connection Block of RetAMPOperatorParm command. 4. Added support for snapshot lists to automation controls. 5. Added support for these MIDI events to automation controls: program change, mono aftertouch, pitch bend. 6. Added snapshot commands. 7. Made note that restoring settings from FLASH/EEPROM or restoring settings to factory default results in the device being reset. 8. Added hardware interface commands for footswitches. 9. Added ability to select multiple MIDI out channels for AMP modify function and to specify the MIDI channel to use for operator inputs that are connected to another operator’s M output. 10. Added mute groups to hardware interface commands. 11. Added support for mute groups to automation controls. 12. Added support for mute groups to footswitches. 13. Added product ID section.
18	16-08-25/ S. Juskiw	Parameters that require the device to be reset before taking effect are indicated in RED underlined text. Search for the text “RED underline” to find the commands that have this requirement.

17	16-08-15/ S. Juskiw	Added AMP commands (advanced MIDI processor).
16	16-05-02/ S. Juskiw	Added automation control commands.
15	15-08-13/ S. Juskiw	1. Added GetMIDIMonitor and RetMIDIMonitor commands. 2. Various changes to support MIDI control ports.
14	15-05-27/ J. Cain	Reformatting from original document.