DXL Host Interface Specifications

Table of Contents

1. Intent and Scope

This document describes the host interface for the DXL. This document has a list of commands and status messages that the DXL supports as well as specifications of the low level protocol for the various DXL hosts. These include the ASCII based hosts (serial and TCP/IP) and register based hosts (Modbus TCP/IP and Omron).

All commands in this document indicate their implementation status. If a command is unimplemented, then it will return a failure if it is received by the DXL. Note that some commands are unimplemented but their corresponding status messages are implemented. The version in which a new command and/or status message is available is described in the implementation status field of each message in the protocol's description.

Some of the host messages may, if they are not implemented, have their syntax changed in subsequent versions of the DXL protocol. The messages whose syntax is subject to change are detailed in each message's implementation status section.

2. Introduction

There are two basic type of hosts, ASCII based and register based. ASCII based hosts communicate with character strings while register based hosts communicate with integers stored in registers. The following sections describe the ASCII and register based hosts. Following these sections is a description of the DXL host protocol (i.e., the command/status messages). The DXL host protocol is common to both ASCII and register based hosts.

In this document, there are references to three basic types of messages: command, status and response. Command messages are defined as messages from the external host to the DXL and status messages are defined as messages from the DXL to the external host. Response messages are defined as messages sent from the DXL to the external host in response to the receipt of a command.

3. ASCII Based Host Support

There are two basic ASCII protocols. Each exchanges a common data format. The only difference is the transport medium. There are protocols for serial and TCP/IP only. The next section describes the format of the ASCII messages and their responses. The last sections detail the variations between the serial and TCP/IP hosts.

3.1.1 ASCII Format

The communication protocol is a simple, ASCII character based one. Command/status messages consist of up to four characters followed by characters and numbers, each separated by one or more spaces[1] from the command/status message and each other. Each message is terminated with an ASCII carriage return. All characters are case insensitive. All numeric parameters are sent as positive, base-ten ASCII strings (i.e., as a sequence of the characters 0-9). Numeric parameters can be padded with zeroes (however they do not get interpreted as octal, as in some programming languages, like C/C^{++} for example). No hexadecimal digits are allowed. Missing parameters are not interpreted as zero (i.e., if a command specifies two parameters, it is not legal to enter one parameter and assume that the others are zero). Extra parameters are ignored if they are included.

The maximum number of characters that can be received by the DXL from the host in one message is limited to 40, not including the carriage return. The maximum message sent out from the DXL to the host is also limited to 40 characters, not including the carriage return. All DXL identifiers are non-negative numbers and the value zero is, generally, reserved for use as a null ID.

An example string is given below:

```
Ical 1 2<CR>
73 99 97 108 32 49 32 50 13     Decimal ASCII equivalents
49 63 61 6C  20 31 20 32 0D     Hexadecimal ASCII equivalents
```

The string sent may have extra characters on the end of the command (i.e., it may have extra spaces or characters after the last number on the line), which are ignored. An extra line feed either before or after the carriage return is also ignored. Therefore the following would be equivalent to the above command:

[1] Spaces and tabs are the only legitimate space characters. Carriage return, form feed and new line are not considered inter-word spaces.

```
Ical    1   2 3 XYZ  <CR><LF>
```

The DXL will respond to any illegal or badly formatted command messages (from the host) with a **Sntx** error message. All commands that are formatted properly will generate either a **Done**, **Busy**, **Fail** or **Sntx** response message followed by the command which caused it. For example, if the **Ical** command was received by the DXL then it could respond one of the following:

```
Done Ical 1 2<CR>
Busy Ical 1 2<CR>
Fail Ical 1 2<CR>
Sntx Ical 1 2<CR>
```

Note that the responses are all formatted (except for the **NOOP**[2]) as if the request was received with correct capitalization and without extra spaces. Therefore the response above would be sent regardless of whether the command received by the DXL was in either format above. The command will always be capitalized exactly as documented in the following sections except, of course, where the command is undefined, in which case the undefined string is returned. Therefore the response to an **ical**, **ICAL** or even **iCaL** would be **Done Ical**. All spaces and/or tabs before the command is eliminated as well. All the numeric parameters will be separated by each other by a single space and will be represented with out leading zeroes. For example, if the command received contained parameters **01 0002** then this would be output as **1 2** in the response.

It is possible to configure the DXL so that the **Done** status is suppressed for all successful command responses. Any command messages which are successful (i.e., did not generate a **Fail**, **Busy**, etc.) would not return the **Done** status response if this was disabled. This can help reduce the amount of traffic between the DCC and the external host.

It is also possible to configure the DXL so that status messages will always be preceded by the **Done** response. Alternatively, it is possible to the configure the DXL so that the responses (i.e., those that are successful) are sent without the **Done** prefix. This can be used to simplify the host programming by reducing the number of messages that need to be parsed, especially if the host doesn't care whether an action was initiated by themselves (response to a command) or the system (status message).

The DXL can be configured so that it requires acknowledgement for any status messages it sends to the host[3]. This is to ensure that the destination received the message (the message could have been lost due to a buffer overflow, for example). For example, when acknowledgement is disabled and a call is connected, the DXL would send the status message **Ical 1 2**<CR> and then the next message (whatever it is). When acknowledgement is enabled then the next message would not be sent until the DXL received `the Ackd Ical 1 2<CR>` response. If the DXL did not receive this response, then it would try to send the `Ical 1 2<CR>` again (maximum of two more retries, for a total of three messages). The amount of time that the DXL will wait before retrying will be specified in the configuration.

---

[2] The **NOOP** command is handled specially for the ASCII based hosts. The received command is not reformatted to its "correct" form. Sending **NoOp Hello there** will result in the exact sentence prefixed with a response message, as in **Done NoOp Hello there**.

[3] The **Ackd** host response (it is never generated by the DXL) is only used for ASCII based hosts because register based hosts have low level protocol acknowledgements that prevent loss of status messages.

If the **Ackd** response is enabled, the host must send the **Ackd** response for any status message received from the DXL. This includes all strings that the host does not recognize. Otherwise the DXL will resend the message which will slow down the regular operation of the system.

Each time that the DXL receives a message from the host, it resets a timer. This timer is programmable in the configuration (i.e., its duration). If the timer expires (i.e., the DXL does not receive a command from the host computer within the timeout period) then an alarm is generated. Typically, the host generates **NOOP** commands to prevent the DXL from timing out, however, any received command (whether legal or not) will reset the timer.

The DXL can be configured to generate a **NOOP** command periodically. The time between **NOOP**s will be set in the configuration. The time between **NOOP**s will be constant (i.e., they are sent regardless of what other commands are sent)[4]. This is typically used by the external host as an assurance that the DXL is running.

3.1.2 Serial

This protocol is implemented on the serial port of the DCC. The serial port is configurable so that hardware and software handshaking can be enabled or disabled, depending upon the user's preference. Hardware handshaking uses the RTS/CTS to determine when the receiver/transmitter can send and receive serial data. Software handshaking uses XON/XOFF characters to prevent character loss by halting the sending of characters until the XOFF condition is cleared. The serial protocol is, otherwise, always active. There are no provisions in the protocol to retransmit missed messages as there is no handshaking (other than the hardware and software handshaking of the serial port).

The number of characters per byte, the number of stop bit, and parity are also configurable on the serial port. Any command messages that have parity errors will result in the DXL responding with an error message to the host. The serial port settings are part of the configuration.

3.1.3 TCP/IP

In this configuration, the DXL transmits the ASCII strings using the TCP/IP protocol over an Ethernet link. TCP/IP guarantees that the individual strings and the messages they contain are received error free and in the order that they were sent.

When using TCP/IP, the DCC operates as a server and the hosts will operate as clients. Since TCP relies on the underlying IP protocol for addressing, each DCC and each host must have a unique IP address (the IP address of each DCC is in the configuration). The DXL uses a user-configurable TCP port number to represent the logical connections to the external host systems.

On system start-up, the DCC(s) monitors the TCP port number for connections from the external host(s). To begin communications the host must initiate a TCP session to the IP address/port number of the DCC. When a session is opened, the DCC determines the host's return IP address and port number

[4] The NOOP is scheduled to occur periodically, however, due to transmission delays, can arrive at a slightly different time. It also depends upon how the user acknowledges the previous commands because the commands are sent sequentially with other commands.

(from the host's initiating connection). All future status messages will be sent through this address and port number. It will also accept and execute commands sent from the host via this connection.

The DCC can open multiple connections per host interface however it can only have one active connection from each source (i.e., each TCP/IP address) at a time. If the DCC is connected to one host and the same host attempts to open a new connection (i.e., from the same IP address) then the existing connection will be terminated and the new connection will be used to send and receive commands. If the DCC is connected to one host and another host attempts to open a new connection (i.e., from another IP address) this connection and the original connection will co-exist at the same time. However, only the last host that sent a command (even an invalid one) will receive status messages. The maximum number of concurrently existing connections is configurable.

4. Register Based Host Support

There are two basic low level protocols supported. One is for Modbus TCP/IP controllers while the other is for Omron controllers. The following section describes the common features of the register based protocols. The last section describes details specific to each protocol.

4.1.1 Registers

The DXL and the external host exchange command and status information by reading and writing blocks of registers. These command and status messages are, generally, associated directly with a particular master station in the DXL. Therefore, the DXL uses a separate set of register blocks for communications with each master station.

In addition to the command registers, each master, station and zone can have "status" registers. These registers (which are optional) can be used to report the current state to the host. These registers are read only registers (from the host's point of view).

4.1.1.1 Input/Output Registers

For each master station controlled by the host, the user allocates (at least) two contiguous blocks of registers[5] for that master to communicate with the external host, plus one additional "handshaking register". One block of registers, called the "output registers", carry status messages from the DXL system to the external host. The DXL will write to these registers and the external host will read them. The other block of registers, called the "input registers", carry command messages from the external host to the DXL system. The external host will write these registers and the DXL will read them. Note, in this document the terms "input" and "output" are with respect to the DXL system.

The external host sends all commands for a particular DXL master station through its input registers. These registers are always located on the DXL pseudo-PLC.

Two different methods can be used to send status messages from the DXL system to the external host. The DXL configuration files determine which method will be used. In the first method, called "peer to

[5] The number of registers allocated per input/output block is fixed at ten registers as is their address. Typically only five registers are needed but more may be better. Their addresses are defined in the configuration (on a per master basis).

peer" mode, the DXL sends status messages to the external host by writing directly to output registers contained within the PLC. In the other method, called "polled" mode, the output registers are located on the DXL "pseudo-PLC", and the external host must read them periodically (i.e., poll), to determine if there is a new message from the DXL system. When polled mode is used, the DXL never writes into the external host. Instead, the external host reads status messages from virtual output registers on the DXL pseudo-PLC.

In peer-to-peer mode, the DXL will always write messages to the output registers, located on the external host, using a block register transfer to update all registers at the same time.

All command and status messages in the register blocks start with a code word followed by parameter words. All codes and parameters are sent as binary words two bytes, or 16-bits, in size. As a result, the code and parameters are limited to the range 0–65535. The value 0 is used as a null code word to indicate that there is no valid message in the register block. The input and output register blocks are written with the code word in the first (lowest addressed) register. The first parameter is in the second register, the second in the third register, etc. The mapping is:

```
Code          →    Base Register
Param #1      →    Base Register + 1
Param #2      →    Base Register + 2
Param #3      →    Base Register + 3
Param #4      →    Base Register + 4
etc.
```

The DXL will process the command message in the input registers when the code register is written. Similarly, it may update the output registers with a new status message when the code register is read. Therefore, if the block of registers can not be read or written in one atomic block transfer operation (i.e. if they are read or written one register at a time), then the registers should be accessed in reverse order (i.e. with the parameter registers being read or written first, and the code register being read or written last). Where possible, block register transfers should be used for better performance.

If the external host sends an invalid register read or write command to the DXL; that is, if it tries to access a register that is not defined in the DXL configuration, or attempts to write to a read only register, the DXL will ignore the command and respond with an error. These errors are logged by the DXL. If the external host issues command messages with invalid or disabled command codes the, DXL will ignore the command and respond with a syntax error message (see **`Sntx`** in the message specifications).

Each time that the DXL receives a message from the host, it resets a timer. This timer is programmable in the configuration (i.e., its duration). If the timer expires (i.e., the DXL does not receive a command from the host computer within the timeout period) then an alarm is generated. Typically, the host generates **`NOOP`** commands to prevent the DXL from timing out, however, any received command (whether legal or not) will reset the timer.

The DXL can be configured to generate a **NOOP** command periodically. The time between **NOOP**s will be set in the configuration. The time between **NOOP**s will be constant (i.e., they are sent regardless of what other commands are sent)[6].

4.1.1.2 Handshake Registers

To ensure that both the DXL and the external host receive every message sent by the other, they must coordinate the message transfers. A "handshake register" and careful sequencing are used to ensure that the sending party does not overwrite any message before the receiving party reads it. This is defined as a single register (defined on a per master basis) in the configuration of the master.

The external host sends command message to the DXL by writing to the input registers. When the external host is writing to the DXL input registers using block transfers, there is no need for handshaking since the DXL will queue each received message for processing and there is no possibility of overwriting the previous register contents.

In peer-to-peer mode the DXL will write a message to the output registers and then wait for the external host to read those registers before writing another message. Since the output registers are in the external host in peer-to-peer mode, the DXL can not tell when the external host has read them. Therefore, the external host must write to the handshake register to acknowledge that it has read the current message from the output registers and that it is ready for the DXL to write the next message (the DXL will not write a new command immediately if it does not have another message to send). The external host can write any value, including 0, into the handshake register[7], it is the write operation that is important, not the value written. If the external host does not update the handshake register, the DXL will not send further status messages to this master until a timeout period (typically 30 seconds, though this may be configurable) has expired.

In polled mode the external host should read the handshake register to check for a new status message. Whenever the DXL has a status message to send it will change the handshake register to return a non-zero value when read. Thus reading the handshake register allows the external host to check for new messages with a single network operation. If a non-zero value is returned, the external host must read the message from the output registers. This will clear the handshake register contents to 0 (unless there is another message immediately ready to be transferred, in which case the next read of the handshake register will return a non-zero value). If there is no message to be sent, then the output code register will be 0.

4.1.1.3 Queue Registers

The DXL also maintains an optional block (for each master) of "queue registers" that show the top entries on the active queue for each master station. Each queue register block requires 2 registers per

[6] The NOOP is scheduled to occur periodically, however, due to transmission delays, can arrive at a slightly different time. It also depends upon how the user acknowledges the previous commands because the commands are sent sequentially with other commands.

[7] Note the handshake register is volatile. This means that the PLC can not read back the value previously written into it. The value returned when reading the handshake register indicates whether the DXL has placed a new message into the output registers.

entry, the first (lowest addressed) register in each pair will hold an alarm code, and the second will hold the ID number of the device that generated the alarm. The length of the queue and its base address is specified in the DXL configuration.

The following table shows the values of the first register (i.e., the alarm code).

Alarm Code	Integer Value
0	No alarm present
1	Station call request
2	Master call request
3	Station audio level alarm
4	Panic alarm call request

4.1.1.4 Master Connection Status Registers

Each master has two (optional) registers to indicate what it is connected to. The first register indicates the type of device the master is connected to. The second register is the ID number of that device. These registers must be contiguous.

The following table shows the mapping of the first register (type of device).

Code	Type of device
0	Not connected
1	Station
2	Master
3	Zone

4.1.1.5 Master Manned/Unmanned Status Register

Each master has a single (optional) register that indicates its current manned/unmanned status. If the master is manned, then this register is set to non-zero. If it is unmanned, then it is set to zero.

4.1.1.6 Station Call Status Register

Each station has a single (optional) register to indicates its call request and connection state. Each bit in the register indicates a different setting. When the bit is on, then that means it is active. Multiple bits can be active at the same time.

0	1	2	3	4	5
Being called	Listening to music	Call request pending	Audio level alarm pending	Being paged	Being monitored

4.1.1.7 Station Enable Status Register

Each station has a single (optional) register to indicate its enable/disable (state). This is a bitmap of the various functions of a station. Multiple bits can be active at the same time.

0	1	2	3
CRQ enabled	Audio level alarm enabled	Music enabled	Tamper enabled

4.1.1.8 Zone Connection Status Register

Each zone will have a single (optional) register that indicates its current connection status. This is a bitmap of the connections which are possible for a zone.

0	1
Being paged	Listening to music

4.1.2 Modbus TCP/IP

Each DCC acts as a server and waits for the external host to connect to it as a client. Since the DXL is acting as a server it does not need to be configured with the IP address and port number of the PLC(s) it will be talking to. The DXL will, instead, obtain this information from the client when it connects. The DXL will then respond to external host via the IP address and port number of the client.

The TCP port number of the DXL server that the external host connects to on the DCC must be configured in the DCC configuration. This TCP port number will be monitored for connections on the DCC.

Only holding register reads and writes are supported and, therefore, all input, output, handshaking and queue registers must be mapped into this area.

Only preset single register (code 6), preset multiple registers (code 16), and read multiple holding registers (code 3) Modbus TCP/IP commands are supported.

The number of active connections allowed depends upon the host protocol in use. If the host is in peer-to-peer mode then the there can only be one active connection per host. If the DCC is connected to one host and another host attempts to open a new connection (i.e., from another IP address and/or port number) then the existing connection will be terminated and the new connection will be used to send and receive commands.

If the host is in polled mode, however, then we can allow multiple active connections. In this situation, we will assume that each connection is concerned with only a specific set of registers that it reads/writes (which correspond to one or more masters). Because the DXL never writes registers on the host, then this will work. The maximum number of master connections which can be open will be limited to 17[8].

4.1.3 Omron

Each DCC sends messages to the Omron PLC via UDP/IP. Messages are formatted as FINS[9] commands. All the registers must be mapped to the data memory (DM) of the PLC. These are type 82 in the OMRON FINS memory map. All other memory area accesses are considered errors.

The port number used for the FINS commands is typically set to 9600. However, the PLC can change the port number so this will be specified in the DXL configuration (if it differs from the default).

[8] There are a maximum of 2 masters per DCC/DCE and 2 microphones per DCC/DCE. Therefore, the maximum number of open connections will depend upon the maximum number of DCE per DCC (currently 4). We also allow one more connection for the master-independent messages.

[9] See "FINS Command Reference Manual" dated June 1993.

Only memory area read, memory area write, memory area fill, multiple memory area read and clock read/write FINS commands are supported. All messages sent to the PLC will request a FINS response. All other messages will return a FINS error.

Due to a quirk in the Omron PLC, the last byte of the IP address (i.e., the 1 in 192.168.0.1) must be the FINS node number. FINS addresses are made up of three parts, the node number, network number and module number. Typically, the module number is always 0 and the network number is always 1. However, it is possible to change these on the PLC; therefore the DCC must make these configurable. The addresses of both PLC and DCC must be specified in the DCC configuration.

All register reads/writes (when the DCC is reading from/writing to memory in the PLC) are sent to the host PLC using the IP address and port number of the PLC. These are formatted as FINS commands. The source and destination address in the FINS command must be configured in the DCC for these updates[10].

The DCC will monitor the UDP port for FINS commands from the PLC (i.e., any packet broadcast to the DCC). The DCC must check the FINS address to determine if the packet was intended for it before decoding it. Any message not addressed to this DCC are ignored.

5. Host Protocol

Responses to commands from the host can be divided into three basic types. There are the good responses, indicating that the operation succeeded. Typically, the good response is to echo the command back with a **Done** command prefix. However, in the DXL configuration, it is possible to suppress the **Done** response when the command is successful. In this case, no response is considered the good response to the command. If an action on the DXL causes a status message to be generated this will be reported to the host. However, if the action is caused by the corresponding host command, then the DXL, typically, will not send the status message[11].

It is also possible to configure the DXL so that status messages will always be preceded by the **Done** response. Alternatively, it is possible to the configure the DXL so that the responses (i.e., those that are successful) are sent without the **Done** prefix. This can be used to simplify the host programming by reducing the number of messages that need to be parsed, especially if the host doesn't care whether an action was initiated by themselves (response to a command) or the system (status message). This can be done to any host port (i.e., ASCII based or register based).

When the DXL receives a unknown or badly formatted command, it responds with the syntax response. This response indicates that the DXL does not understand the command. These are considered syntax errors and illicit a **Sntx** response. Similarly if the command can not be performed (i.e., say due to a hardware error) this will result in a **Fail** response.

[10] When the DCC is sending a FINS response to a PLC FINS command, the source and destination address of the PLC FINS command can be swapped to build the response's addresses. When the DCC is sending commands to the PLC, however, it must use the configured FINS addresses (which must be specified for both the DCC and PLC in the DCC).

[11] There are exceptions to this rule. They are noted in the command summaries, where appropriate, below. Basically this says that if you send a command to a host, that host doesn't get a status message, just a response. Other hosts (i.e., not the one the command was sent to) will receive a status message to update their state if applicable.

The only other response is the **Busy** response, which is restricted to failed call commands.

Note that not all commands will return a **Done** message if successful (regardless of the above settings). Each command's responses are different. The **Sntx** response, on the other hand is always present for any command which has syntax problems. These can include permission problems (i.e., a master can not call a station if the station is not on its permission list ergo this is considered a syntax problem).

The DXL can individually disable any or all of these commands/status messages in the DXL Administrator[12]. If the command message is disabled, then the status message is disabled and vice versa. Any command sent which is disabled will return the **Sntx** response. Status messages that are normally generated will be suppressed (i.e., not sent to the external host) when they are disabled. This can be used to reduce the traffic between the DCC and the external host, especially for status messages which the external host is not going to parse anyways.

One DCC can have multiple host ports running at the same time (i.e., two or more host ports on a single DCC). These can operate independently and typically service different sets of masters on a DCC. In most commands the master to which the command is directed is part of the message syntax. Some commands, however, do not involve masters. These commands for these are should be sent to the default host (defined in the configuration for each host port) though they can be sent to any host which controls that device.

In the following sections, the host interface messages are described individually. Examples of each command, response and status message are provided as well.

5.1 Compatibility with DXI

The DXL host protocols are essentially new protocols. They are based on Microcomm DXI protocols developed by Harding Instrument Co. Ltd., hence is a overlap between the host messages (i.e., most ASCII strings and register functions are the same). There are, however, some significant differences between the two protocols which are noted in the command summaries below. This section describes what has to be done to make a DXL system look more like a DXI system so that the user does not have as many changes to make to implement a DXL intercom system.

Note that there are limits to how much the DXL can be configured so that it looks more like a DXI system. Common functions like call requests, calls, etc. use the same syntax as the DXI. There are certain functions, however, that only a DXL can do or only a DXI can do. Therefore it is impossible to make the two systems look, from the host's point of view, exactly the same. However, while complete compatibility is not possible, it is possible to make a DXL system look more like a DXI system.

There are two ways to make a DXL and DXI driver that are essentially equivalent (i.e., so that the same host driver can be used for the DXL and DXI). The first is to change an existing DXI driver to handle the new DXL commands. This is fairly simple. The second thing that can be done is to disable the new DXL commands which will cause the DXL to send the equivalent DXI commands. How to do this is described in the following sections.

[12] In the "Host Port" properties "Messages" tab.

5.1.1 Changing DXI Driver To Handle New DXL Commands

If the user wishes to change their DXI driver to handle new DXL commands, then the following status messages (i.e., messages sent from the DXL to the host computer) must be made equivalent (i.e., their function codes can be treated as the same):

DXI command	DXL equivalents
EndC	Bend Iend Mend Pend Vend
AdMS	AdMG
EnbM	EnGM
Enbs	EnGS
Levl	GLev
RmMS	RmMG
SetM	SetG
Smas	Gmas
SMus	GMus

If the user wants to change their existing DXI host to accept these equivalent commands (i.e., accept either an **AdMS** or **AdMG** as the same status message/response) then there is no need to disable these commands. This probably just involves doing an *if* function code == **AdMS** *or* function code == **AdMG** etc. Note that the **EndC** function code is equivalent to 5 different function codes except that there is an extra parameter for the DXL equivalents. This is the specific station, visiting booth, master or zone that has ended. This can be ignored if these are treated as equivalent to the **EndC** function code.

In order to allow the DXL to emulate a DXI host better, we have also extended the above function codes (i.e., the station specific ones) on the DXL so that they will, if there is no station with a particular number, look to see if there is a station group with the same number. If there is then the function code will act like the corresponding group command. For example, if there is a station group 1000 but no station 1000 then doing a "**SetM** 1000 1" is the same as doing a "**SetG** 1000 1". This change has been made to the DXL (see the descriptions of the above functions) so that all the station group specific commands do not need to be used as long as there is no overlapping of station and station group numbers.

This is obviously the simplest way to make the DXL and DXI more compatible. While this is probably preferred, there may be cases where it is not necessarily easy to change an existing DXI driver. Therefore, the user will have to change the DXL configuration to eliminate the new DXL commands.

5.1.2 Changing DXL System To Eliminate New DXL Commands

In order to eliminate the new DXL commands and revert to the equivalent DXI commands, the user has to disable some of the function codes in the host's configuration (i.e., in the DXL Administrator). The following two sections describe which messages have to be disabled. Note that this must be done for each host in the configuration.

5.1.2.1 Bend, Iend, Mend, Pend, Vend

Typically, ending a call from a master to a station, visiting booth, zone or master is done by sending an **EndC** command to the host. However, the user can implicitly end the current connection by sending a new **Bcal**, **Ical**, **Mcal**, **Page** or **Vcal** command to a different destination device. The current call will be

disconnected and the host informed via a status message. If the call being implicitly disconnected is to a station, for example, then the host will be sent a **Iend** message from the host to indicate which a call to a particular has been ended. In the DXI, however, there is no **Iend** message so the DXI returns an **EndC** status message to the host. By disabling the above function codes in the DXL, the DXL can be made to send the **EndC** status message exclusively.

Note that disabling these commands can make the implementation of the host more complicated. The above function codes tend to simplify the host programming by telling the host exactly which station, visiting booth, zone or master was disconnected. The **EndC** function code has no such indication. Therefore, using the host must, upon receiving the **EndC** command, clear all devices that the master could have been connected to.

For example, if a call to station 1000 was implicitly ended by a new connection then the DXL would send either an "**Iend** 1 1000" or "**EndC** 1". In the first case the host would just have to clear master 1's connection to station 1000 while in the second all possible connections from master 1 would have to be cleared (or the host would have to keep track of its current connection, not necessarily an easy thing to do).

5.1.2.2 AdMG, EnGM, EnGS, GLev, RmMG, SetG, GMas, GMus

These function codes are used by the DXL to deal with station groups. In the DXL station groups are in a distinct user identifier space from the stations. In the DXI, stations and station groups are in the same user identifier space. As a result, in the DXL we can have a station group with the same number as a station while in the DXI we can not. This meant that we had to have separate host commands for stations and station groups. The above are the station group commands that we added to replace/differentiate the following station commands (in order)

AdMS, EnbM, Enbs, Levl, RmMS, SetM, SMas, SMus

In order to allow the DXL to emulate a DXI host better, we have extended the above function codes (i.e., the station specific ones) on the DXL so that they will, if there is no station with a particular number, look to see if there is a station group with the same number. This makes it so that the user does not have to send these new host commands (i.e., the station group ones) as long as there is no overlap between the stations and station groups.

Status messages related to the station groups will be reported with the group function code if these are not disabled. For example, if the user added a group to a master monitor list using "**AdMS** 1 1000" then the DXL will report (when the host sends a "Stat 1") "**AdMG** 1 1000" if the **AdMG** function code is not disabled. If this function code is disabled then the DXL will respond with an "**AdMS** 1 1000" which is similar to what the DXI does.

Note that for this to work, the user must take care not to have any stations or station groups with the same number. There is no way for the DXL to check this so it is up the user to carefully number their stations and station groups to prevent overlap.

6. Register Function Codes

The following table can be used to decode register function codes into ASCII messages. The page number listed is the page where that command is located (this is also a hyperlink to that page so the clicking on the page number or ASCII code will take you to it).

The ASCII strings and register function codes (i.e., the number used when a register based protocol, as opposed to an ASCII based protocol, is used) are in some cases reused from the DXI host protocol. Messages that were part of the DXI host protocol and have been reused (though there syntax and usage may be slightly different in the DXL than the DXI) for the DXL are have been highlighted in yellow. New DXL messages are not highlighted.

Register Code	ASCII Code	Page
1	Icrq	62
2	Ican	61
3	Mcrq	73
4	Mcan	72
5	Date	32
6	Time	115
7	Ical	60
8	Mcal	71
9	Page	90
10	EndC	49
11	Enbl	43
12	Stat	109
15	Done	34
16	Busy	28
17	Fail	54
19	Halm	59
20	Cflt	30
21	Next	88
22	IVad	68
23	MVad	87
24	Sgnl	104
25	EndS	50
28	Enbs	45
30	NOOP	89
33	SetO	102
43	SetS	103
44	SetM	101
45	Alvl	21
46	Acan	16
47	Levl	69
48	Imon	64
51	EndM	48
57	PAlm	91
59	MusM	84
60	MusN	85
61	MusP	86
62	ZMus	123

Register Code	ASCII Code	Page
63	ZMas	122
64	SMus	106
65	SMas	105
66	Vset	118
67	Vstp	120
68	VstB	119
69	Vcal	114
70	Vmon	116
72	EnbM	44
90	AdMS	19
91	RmMS	99
92	PVad	96
93	IRec	65
94	VRec	117
97	MRec	83
98	EndR	49
103	Iset	68
104	Istp	67
113	MDst	77
114	MDup	78
115	MDdn	74
200	Iend	63
201	Mend	79
202	Pend	94
203	AllS	20
204	Sntx	107
205	Ackd	17
206	Hack	58
207	EnbH	42
208	Cack	29
209	EnbC	38
210	EnbE	40
212	Talm	111
213	Tack	110
214	Tcan	112
215	EnbT	46
216	Dflt	33

Register Code	ASCII Code	Page
217	Dack	31
218	EnbD	39
219	Mflt	80
220	Mack	70
221	EnbF	41
222	EnGS	52
223	Eflt	37
224	EnGM	51
225	EnGT	53
227	GLev	55
228	GMus	57
229	GMas	56
230	Eack	36
231	Bset	26
232	Bstp	27
233	MDon	76
234	MDof	75
235	Vend	115
236	Bcal	22
237	Bend	23
238	Bmon	24
239	AdMG	18
240	RmMG	97
241	BRec	25
242	Zadd	121
243	Zsub	126
244	Zset	124
245	Zstp	125
246	SetG	100
247	Pcrq	93
248	Pcan	92
249	Mptt	82
250	Mmut	81
251	Dvol	35
252	Priv	95
253	Sadd	99
254	Ssub	108

Acan – Audio Level Alarm Cancel

Register Function Code: 46

Parameters:

Master
Station

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.5.

Command/Response Messages:

Tells "Master" to remove the audio level alarm from "Station" from its active or acknowledged queue. This will return a good response if the "Station" even if the "Master" did not have an active or acknowledge alarm.

Examples:

Acan 10 1130 *Remove the level alarm of station 1130 from master 10's queues*

Status Messages:

An audio level alarm from "Station" and been removed from at "Master". This could have been on either the active or acknowledged queue of "Master".

Examples:

Acan 10 1130 *Level alarm of station 1130 removed from master 10' s queues*

Ackd – Acknowledgement of Status Message (From Host)

Register Function Code: 205

Parameters:

Function Code and Parameters.

Implementation Status:

Response messages implemented in DXL version 0.0.1b1.

Host Response Messages:

This acknowledgement response indicates that a status message was received by the host successfully. The parameter string should be an echo of the status message which was received and is sent from the host to the DXL. The **Ackd** response message can be disabled in the configuration to reduce the amount of message traffic if positive confirmation is not required. The exact syntax of the command must be preserved except for spacing between parameters which can be different than the status message and capitalization of the **Ackd** and status message. This is only used for ASCII based hosts.

Examples:

Ackd Ical 10 1130 *Host acknowledges receipt of Ical 10 1130 command.*

Status Messages:

N/A.

AdMG – Add Station Group to Intercom Auto-Monitor List

Register Function Code: 239

Parameters:

Master
Station Group

Implementation Status:

Command/response and status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

This command is sent to monitor a "Station Group" from "Master". The "Station Group" members are added to the master's monitor list (which is cycled through on the "Master") instead of doing a dedicated monitor as the **Imon** command does. This can not be passed with 0 as the "Station Group", it must be passed a valid station group number. Returns a Done response if the command was successfully received.

Examples:

AdMG 1 99 *Add station group 99's members to master 1's monitor list.*

Status Messages:

Sent when the monitor list of a master is changed from within the DXL system. This can also be sent in response to a **Stat** or **AllS** command.

Examples:

AdMG 1 99 *Master 1 add station group 99's members to its monitor list.*

AdMS – Add Station to Intercom Auto-Monitor List

Register Function Code: 90

Parameters:

Master
Station

Implementation Status:

Command/response and status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

This command is sent to monitor a "Station" from "Master". The "Station" is added to the master's monitor list (which is cycled through on the master) instead of doing a dedicated monitor as the **Imon** command does. This can not be passed with 0 as the "Station", it must be passed a valid station number. Returns a good response if the command was successfully received.

DXL version 3.0.1b3: If there is no station with this number then the "Station" number will be looked up to see if there is a station group which has this number. If there is then this will add the station group's members to the master's monitor list, as if the user entered the **AdMG** command with the "Master" and "Station" as the parameters.

Examples:

AdMS 1 99 *Add station 99 to master 1's monitor list.*

Status Messages:

Sent when the monitor list of a master is changed from within the DXL system. This can also be sent in response to a **Stat** command.

Examples:

AdMS 1 99 *Master 1 add station 99 to its monitor list.*

AllS – Request All Master/Paging Microphone Status

Register Function Code: 203

Parameters:

Implementation Status:

Command/response messages implemented in DXL version 0.1.0b1.

Command/Response Messages:

This tells the host to report the current status of all the masters and paging microphones assigned to it. This also includes any global information (i.e., which is not specific to one particular master or paging microphone). This is exactly like generating a **Stat** command for all the masters assigned to the host.

Examples:

AllS *Report the status of anything assigned to this host.*

Status Messages:

N/A.

Alvl – Audio Level Alarm Occurred

Register Function Code: 45

Parameters:

Master
Station

Implementation Status:

Command/response and status messages implemented in DXL versions 0.5.5.

Command/Response Messages:

Like call requests, this message can be sent from the host to generate an audio level alarm on "Station" (i.e., to act like an audio level alarm has occurred). Like call requests, this alarm will be queued on the master of "Station" (not necessarily on the master passed in the command). This never sends a good response and may (if the station's audio level alarms are disabled) not queue the alarm at the station's master. If this command is successful, then the status message is generated to indicate that the command has been processed.

Examples:

Alvl 10 1130 *Generate an audio level alarm from station 1130*

Status Messages:

Reports that an audio level alarm has occurred. This can also be sent in response to a **Stat** or **AllS** command (if there are active audio level alarms). Only those alarms which are active will be reported in the **Stat** or **AllS** command.

Examples:

Alvl 10 1130 *An audio level alarm from station 1130 has been queued on master 10*

Bcal – Visiting Booth Call[13]

Register Function Code: 236

Parameters:

Master
Visiting Booth

Implementation Status:

Command/response messages implemented in DXL version 2.0.1b3.

Command/Response Messages:

Attempts a call from "Master" to "Visiting Booth". Specifying a "Visiting Booth" of 0 will disconnect the current call (see **Iend/Mend/Pend**). If "Master" is in a call to something else (i.e., a master, station, etc.) then this will disconnect that call (and generate an **Iend/Mend/Pend** for "Master") before trying to connect to "Visiting Booth". The **Iend/Mend/Pend**, in this case, will always be sent before the response. If "Master" was doing a monitor, then that will be disconnected (and generate an **EndM** for "Master") before trying to connect to "Visiting Booth". The **EndM**, in this case, will always be sent before the response. Specifying "Master" as 0 is a syntax error. The good response will be generated if the connection is made or the device to which "Master" was previously calling was "Visiting Booth". In the second case, the disconnection is not done and "Master" and "Visiting Booth" are left connected. If the "Visiting Booth" is busy in a higher priority call, then this command will generate a busy response.

Examples:

Bcal 10 1130 *Call visiting booth 1130 from master 10.*

Status Messages:

Reports that "Visiting Booth" has been connected to "Master". This can also report what station (if any) "Master" is connected to when the **Stat** or **AllS** command is sent. If no station is connected to this master, then the message is not generated (i.e., is not sent with a station of 0).

Examples:

Bcal 10 1130 *Visiting Booth 1130 was called from master 10.*

[13] This is similar to **Vcal** except that this command is for pre-defined (i.e., in configuration) visiting booths, not dynamic ones (i.e., host defined).

Bend – End Call To Visiting Booth[14]

Register Function Code: 237

Parameters:

Master
Visiting Booth

Implementation Status:

Command/response and status messages implemented in DXL version 2.0.1b3.

Command/Response Messages:

This command is used to end the call on the "Master". Specifying a "Master" of 0 is a syntax error. If "Visiting Booth" is 0, then any existing visiting booth call will be ended (if there was a page or master call, then this would not be disconnected and this would return a failure). If "Visiting Booth" is not 0, then "Master" will only be disconnected if it is connected to "Visiting Booth". The response will be generated even if this master was not previously connected to anything (i.e., it returns a good response whether it actually does any disconnects or not). This will only end calls, not monitors.

Examples:

Bend 10 0 *Ends current call, if any, on master 10*
Done Bend 10 30 *Response to above command if connected to visiting booth 30*
Bend 10 30 *Ends current call on master 10 but only if from visiting booth 30*
Done Bend 10 30 *Response to above command if connected to visiting booth 30*

Status Messages:

When a new connection is made without ending the previous one (i.e., suppose the user does two consecutive **Bcal** to different stations) a status message will be generated. This will be generated to tell the host which visiting booth call has been ended. This will not be generated if the call ended is not to a visiting booth. If there is an **EndC** used to end the call this will not be generated.

Examples:

Bend 10 1130 *Current call from master 10 to visiting booth 30 was ended*

[14] This is similar to **Vend** except that this command is for pre-defined (i.e., in configuration) visiting booths, not dynamic ones (i.e., host defined).

Bmon – Monitor Visiting Booth From Master[15]

Register Function Code: 238

Parameters:

Master
Visiting Booth

Implementation Status:

Command/response and status messages implemented in version 2.0.0.

Command/Response Messages:

Sent to tell "Master" to monitor "Visiting Booth". If "Visiting Booth" is 0, then "Master" will stop monitoring whatever it was previously (if anything) monitoring, regardless of whether it was a station or not. This is the same as the **EndM** command. If "Master" was not monitoring anything, then this will result in a good response. If "Visiting Booth" is not 0, then this "Master" will monitor "Visiting". If "Master" was already monitoring "Visiting Booth" then this will result in a good response but the connection will be left. If "Visiting Booth" is not 0 and this master was monitoring some other device, this device will be disconnected (which will result in an **Bmon/Vmon/Imon X 0** status message being sent before the **Bmon** response is generated).

Examples:

Bmon 10 1130 *Monitor visiting booth 1130 on master 10*
Bmon 10 0 *Stop monitoring on master 10*

Status Messages:

Reports that "Master" is now monitoring "Visiting Booth". This can also be sent in response to a **Stat** or **AllS** command (if a monitor was in progress). This is only reported (in response to the **Stat** or **AllS** command) if the visiting booth being monitored is not 0.

Examples:

Bmon 10 1130 *Master 10 is now monitoring visiting booth 10*

[15] This is similar to Vmon except that this command is for pre-defined (i.e., in configuration) visiting booths, not dynamic ones (i.e., host defined).

BRec – Record Visiting Booth[16]

Register Function Code: 241

Parameters[17]:

Call Recorder
Visiting Booth

Implementation Status:

Command/response messages implemented in DXL version 2.0.0.

Command/Response Messages:

Connects "Visiting Booth" to the "Call Recorder". "Visiting Booth" must be defined in the configuration for this to be used. "Call Recorder" can not be 0. If the "Visiting Booth" is 0, then the call recorder is disconnected (from whatever it was connected to, even if that was not a visiting station, exactly like doing an **EndR**). If "Visiting Booth" was already connected to the "Call Recorder", then this will return a good response (if "Call Recorder" is not 0). If "Visiting Booth" is not connected (and "Call Recorder" is 0) then this will also return a good response. If "Call Recorder" was connected to any other device, then this will be disconnected and an **EndR** command will be generated before this command's response is received by the host.

Examples:

BRec 1 10 *Record visiting booth 10 on call recorder 1*

Status Messages:

Reports that "Visiting Booth" is being recorded by "Call Recorder". This can be also be sent in response to a **Stat** or **AllS** command. Only call recorders which are recording stations will be sent this messages in response to the **Stat** or **AllS** command.

Examples:

BRec 1 10 *Visiting booth 10 is being recorded by call recorder 1*

[16] This is similar to **VRec** except that this command is for pre-defined (i.e., in configuration) visiting booths, not dynamic ones (i.e., host defined).
[17] The parameters for this command differ from the corresponding DXI command.

Bset – Start Visiting Booth (With/Without Timeout) [18]

Register Function Code: 231

Parameters:

Master
Visiting Booth
Timeout (minutes)

Implementation Status:

Command/response and status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

Activate a "Visiting Booth" connection. If "Timeout" is 0 then the visiting booth connection will not timeout (i.e., it can last forever if necessary). If "Timeout" is not 0, then this will limit the connection to a maximum of that length of time (the connection could be prematurely ended by one of the visiting stations hanging up, if they have handsets). If "Visiting Booth" was already active, then this command will return a good response (as if they had just been made active) but will reprogram the timer to the value passed in the new command. If either or both stations are busy, a busy response will be generated. If the visiting stations have handsets then the connection will not be made until they go off hook (if they were both idle and off hook at the time the command was sent, then the connection is made immediately). The 'Master' can be specified as 0 or any master assigned this host.

Examples:

Bset 10 5 0 *Start never-ending visiting booth 5 connection from master 10*
Bset 0 5 20 *Start 20 minute visiting booth 5 connection from no master*

Status Messages:

This reports that a visiting booth call has started. This can also be generated by a **Stat** or **AllS** command. Only visiting booths that are active will be reported in response to the **Stat** or **AllS** command. The amount of time left (rounded up to the nearest minute) will be sent in this case.

Examples:

Bset 10 5 0 *A never-ending visiting booth 5 connection was started by master 10*
Bset 0 5 20 A *20 minute call visiting booth 5 call was started by no master*

[18] This is similar to **Vset** except that this command is for pre-defined (i.e., in configuration) visiting booths, not dynamic ones (i.e., host defined).

Bstp – Stop Visiting Booth Connection[19]

Register Function Code: 232

Parameters:

Master
Visiting Booth

Implementation Status:

Command/response and status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

Sent from the host to deactivate "Visiting Booth". If "Visiting Booth" was not active, then this will return a good return code (i.e., it acts like it was just disconnected). If the "Stations" were in visiting booth connections with other stations (and not each other) then a fail response will be generated. The 'Master' can be specified as 0 or any master assigned this host.

Examples:

Bstp 10 5 *Stop visiting booth 5's connection from master 10*

Status Messages:

Reports when "Visiting Booth" is disconnected. This typically happens when the timeout occurs.

Examples:

Bstp 10 5 *Visiting booth 5's connection has ended (i.e., timed out or hung up).*

[19] This is similar to **Vstp**except that this command is for pre-defined (i.e., in configuration) visiting booths, not dynamic ones (i.e., host defined).

Busy – Busy Status

Register Function Code: 16

Parameters:

Function Code and Parameters.

Implementation Status:

Response message implemented in DXL version 0.1.0b1.

Response Message:

This response message indicates that the destination for a call command was busy in a call with a higher priority device. The parameter string is an echo of the command which caused the problem..

Examples:

Busy Ical 10 1130 *Station 1130 is busy and can not be called from master 10*

Status Messages:

N/A.

Cack – Communication Fault Acknowledge

Register Function Code: 208

Parameters:

Master
Exchange

Implementation Status:

Command/response and status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

This is sent to acknowledge a communication (i.e., Ethernet) alarm on the DCC/DCE "Exchange" by "Master" (i.e., to move it from active to acknowledged queue). If the alarm is already on the acknowledged queue, then this will return a good response. If the alarm is not active or already acknowledged then this will return a fail response. This can be disabled by use of the **EnbC** host command.

Examples:

Cack 10 1000 *Acknowledge communication fault on exchange 1000 from master 10*

Status Messages:

This status message is sent when a communication alarm from the DCC/DCE "Exchange" is acknowledged on "Master". This can be generated due to a **Stat** or **AllS** command on the master. Only exchanges which are acknowledged on this master will be reported in response to the **Stat** or **AllS** command.

Examples:

Cack 10 1000 *Communication fault on exchange 1000 acknowledged on master 10*

Cflt – Communication Fault

Register Function Code: 20

Parameters:

Master
Exchange
State (0/1)

Implementation Status:

Status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

N/A.

Status Messages:

This is sent when communication alarms[20] between DCCs (i.e., exchanges) occur (this is, obviously not applicable to single exchange DCC systems). If the third parameter is set to 1 (one), then a fault has been generated and queued at "Master" (the master will be the same master where trouble alarms on the local exchange are reported, not where "Exchange" reports its trouble alarm to as that is the one we can not communicate with). If the third parameter is a 0 (zero), then a fault has been cleared from "Master". This can be generated due to a **Stat** or **AllS** command on the master. Only exchanges which have active, unacknowledged communication faults will be reported in response to the **Stat** or **AllS** command.

Examples:

Cflt 10 1000 1 *Fault detected on exchange 1000 was queued on master 10*
Cflt 10 1000 0 *Fault cleared on exchange 1000 removed from master 10*

[20] The faults which can generate this are Ethernet failures..

Dack – DCC/DCE Trouble Acknowledge

Register Function Code: 217

Parameters:

Master
Exchange

Implementation Status:

Command/response messages implemented in DXL version 0.5.3b3.
Status message implemented in DXL version 0.5.3b3.

Command/Response Messages:

This is sent to acknowledge a trouble alarm on the exchange's DCC and/or DCEs by "Master" (i.e., to move it from active to acknowledged queue). The "Exchange" is the exchange (i.e., from the configuration) whose DCC and/or DCE reported trouble. If the alarm is already on the acknowledged queue, then this will return a good response. If the alarm is not active or already acknowledged then this will return a fail response.

Examples:

Dack 10 1 *Acknowledge hardware trouble on DCC 11 on master 10*

Status Messages:

This status message is sent when a hardware alarm from the exchange's DCC and/or DCE is acknowledged on "Master". This can be generated due to a **Stat** or **AllS** command on the master. Only masters who have acknowledged the alarm will be reported in response to the **Stat** or **AllS** command.

Examples:

Dack 10 1 *Trouble detected on DCC 1 was acknowledged on master 10*

Date – Set System Date

Register Function Code: 5

Parameters:

Year (2000-2999)
Month (1-12)
Day (1-31)

Implementation Status:

Command/response messages implemented in DXL version 0.5.3b3.
Status message implemented in DXL version 0.5.3b3.

Command/Response Messages:

This command is used to set the date on the DXL's real time clock. The year must be given in full 4 digit format. The month is a number from 1 to 12 with 1 corresponding to January. The day is a number from 1 to 31. The intercom system checks the number of days in the month to ensure that it is correct. Any errors in the format of the date is considered a syntax error. Otherwise, if the clock is changed, then a good response will be sent.

Examples:

Date 2002 12 25 *Set the date to 2002/12/25.*

Status Messages:

This message will reports that the DXL's clock has changed. When the clock is set on another host and it propagates to the local host, then this message will be generated by the local host (i.e., if there are multiple host ports defined then sending a **Date** command on one will result in **Date** status messages to all the others). However, if the clock was changed due to a **Date** command on this host, then this status message will not be generated, just the response.

Examples:

Date 2002 12 25 *Date was changed to 2002/12/25.*

Dflt – DCC/DCE Trouble Alarm

Register Function Code: 216

Parameters:

Master
Exchange
Status (0/1)

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.3b3.

Command/Response Messages:

N/A.

Status Messages:

This status message is sent when a 'trouble' alarm on an exchange's DCC and/or DCEs is queued at "Master" or when it is cancelled from "Master". The "Exchange" is the exchange number (i.e., from the configuration) where the trouble is occurring. If "Status" is set to 1 (one), then the trouble has been detected and queued on "Master". If "Status" is a 0 (zero), then the trouble has been cancelled from "Master" (typically due to the source of the trouble(s) being cleared). This can be generated due to a **Stat** or **AllS** command on the master. Only exchanges whose DCC and/or DCEs that have unacknowledged faults are reported in response to the **Stat** or **AllS** command. These trouble alarms can be enabled/disabled by the host via the **EnbD**host command.

Examples:

Dflt 10 1 1 *Hardware trouble detected on DCC 1 is queued on master 10*
Dflt 10 1 0 *Hardware trouble cleared on DCC 1 removed from master 10*

Done – Command Successfully Completed Status

Register Function Code: 15

Parameters:

Function Code and Parameters.

Implementation Status:

Response messages implemented in DXL version 0.1.0b1.

Response Messages:

This response message indicates that a command was executed successfully. The parameter string is an echo of the command which completed. This status message can be disabled in the intercom system configuration to reduce the amount of message traffic if positive confirmation is not required. This is typically used to indicate a successful execution of a command.

Examples:

Done Ical 10 1130 *The command "Ical 10 1130" was successful*

Status Messages:

N/A.

Dvol – Master Destination Volume Adjustment

Register Function Code: 251

Parameters:

Master
Postive
Volume setting

Implementation Status:

Command/response messages implemented in DXL version 3.0.0b4.

Command/Response Messages:

This command changes the volume of the station, visiting booth, or page zone that "Master" is connected to. If "Master" is connected to another master then this will have no effect though it will return a good return code. If there is no connection currently in progress then this has no effect though it will still return a good return code. This must be sent after the connection of the master has been made. The fail response only indicates an error in the syntax of this command.

The second parameter, "Positive" is combined with the third parameter, "Volume setting", to get the absolute value to set the destination volume multiplier. If the "Positive" parameter is 0 then the "Volume setting" is negative. If the "Positive" parameter is 1 then the "Volume setting" is positive. "Volume setting" refers to dB gain and must be 16 or less. This setting is not "sticky" in the sense that when the master's current connection is ended the next call to this same device or any other device will revert to 0 dB gain.

Examples:

Dvol 10 1 8 *Set whatever master 10 is connected to have 8 dB gain.*
Dvol 10 0 16 *Set whatever master 10 is connected to have -16 dB gain.*

Status Messages:

N/A.

Eack – External Host Fault Acknowledge

Register Function Code: 230

Parameters:

Master
Host
State (0/1)

Implementation Status:

Command/response and status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

This is sent to acknowledge a communication alarm on the “Host” by “Master” (i.e., to move it from active to acknowledged queue). If the alarm is already on the acknowledged queue, then this will return a good response. If the alarm is not active or already acknowledged then this will return a fail response.

Examples:

Eack 10 1000 *Acknowledge communication fault on host 1000 from master 10*

Status Messages:

This status message is sent when a communication alarm from the host port “Host” is acknowledged on “Master”. This can be generated due to a **Stat** or **AllS** command on the master. Only exchanges which are acknowledged will be reported in response to the **Stat** or **AllS** command.

Examples:

Eack 10 1000 *host fault on host 1000 acknowledged on master 10*

Eflt – External Host Fault

Register Function Code: 223

Parameters:

Master
Host
State (0/1)

Implementation Status:

Command/response and status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

N/A.

Status Messages:

This is sent when communication alarms between the DCC and the external host occur[21]. If the third parameter is set to 1 (one), then a fault has been generated and queued at "Master". If the third parameter is a 0 (zero), then a fault has been cleared from "Master". This can be generated due to a **Stat** or **AllS** command on the master. Only exchanges which have active, unacknowledged communication faults will be reported in response to the **Stat** or **AllS** command. These can be enabled/disabled with the **EnbE** host command.

Examples:

Eflt 10 1000 1 *Fault detected on host 1000 was queued on master 10*
Eflt 10 1000 0 *Fault cleared on host 1000 removed from master 10*

[21] These are communication faults like serial or Ethernet faults, typically. They occur if the DCC has not received any messages from the external host for a configurable amount of time. This is essentially an inactivity timeout.

EnbC – Enable/Disable Communication (Ethernet) Fault

Register Function Code: 209

Parameters:

Master
Exchange
Status (0/1)

Implementation Status:

Command/response and status messages implemented in DXL version 2.0.1b3.

Command/Response Messages:

Enables or disables faults from “Exchange” on the queue of “Master”. The “Status” is a 1 (one) to enable the fault and a 0 (zero) to disable the fault. If the fault is already enabled/disabled by on “Master”, then this will return a good response. If the fault can not be enabled/disabled by “Master”, then the fail response is returned. This is used to enable/disable **Cack**faults.

Examples:

EnbC 10 1000 0 *Request to disable fault of exchange 1 from master 10*

Status Message:

Reports when all faults from “Master” are disabled. This can be generated by the **Stat** or **AllS** command as well. Only communication faults which are disabled are reported in response to the **Stat** or **AllS** command.

Examples:

EnbC 10 1000 1 *Master 10 has enabled exchange 1’s fault*

EnbD – Enable/Disable DCC/DCE Trouble Alarm

Register Function Code: 218

Parameters:

Master
Exchange
Status (0/1)

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.3b3.

Command/Response Messages:

Enables or disables faults on “Exchange” on the queue of “Master”. The “Exchange” is the DCC/DCE exchange number (i.e., from the configuration). The “Status” is a 1 (one) to enable the trouble alarm fault reporting and a 0 (zero) to disable the fault reporting. If the trouble alarm is already enabled/disabled by on “Master”, then this will return a good response. If the fault can not be enabled/disabled by “Master”, then the fail response is returned. This can be used to enable/disable **Dflt**fault reports.

Examples:

EnbD 10 1 0 *Request to disable fault of DCC 1 from master 10*

Status Message:

Reports when faults from “Master” are enabled/disabled on the DXL system (i.e., not from the host). This can be generated by the **Stat** or **AllS** command as well. Only an exchange whose DCC/DCEs that have disabled faults are reported in response to the **Stat** or **AllS** command.

Examples:

EnbD 10 1130 1 *Master 10 has enabled DCC 1’s fault*

EnbE – Enable/Disable External Host Fault

Register Function Code: 210

Parameters:

Master
Host
Status (0/1)

Implementation Status:

Command/response and status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

Enables or disables faults from "Host" on the queue of "Master". The "Status" is a 1 (one) to enable the fault and a 0 (zero) to disable the fault. If the fault is already enabled/disabled by on "Master", then this will return a good response. If the fault can not be enabled/disabled by "Master", then the fail response is returned. This can be used to enable/disable the reporting of **Eflt** external host faults.

Examples:

EnbE 10 1000 0 *Request to disable fault of host 1 from master 10*

Status Message:

Reports when faults from "Master" are enabled/disabled on the DXL system (i.e., not from the host). This can be generated by the **Stat** or **AllS** command as well. Only communication faults which are disabled are reported in response to the **Stat** or **AllS** command.

Examples:

EnbE 10 1000 1 *Master 10 has enabled host 1's fault*

EnbF – Enable/Disable Master Hardware Fault

Register Function Code: 221

Parameters:

Master
Master
Status (0/1)

Implementation Status:

Command/response and status messages implemented in DXL version 2.0.1b3.

Command/Response Messages:

Enables or disables faults on the first "Master" on the queue of the second "Master". The "Status" is a 1 (one) to enable the fault and a 0 (zero) to disable the fault. If the fault is already enabled/disabled by on the second "Master", then this will return a good response. If the fault can not be enabled/disabled by the second "Master", then the fail response is returned. This is used to enable/disable the master hardware fault alarm reporting (see **Mflt**).

Examples:

EnbF 10 2 0 *Request to disable fault of master 2 from master 10*

Status Message:

Reports when faults from "Master" are enabled/disabled on the DXL system (i.e., not from the host). This can be generated by the **Stat** or **AllS** command as well. Only masters that have disabled faults reported in response to the **Stat** or **AllS** command.

Examples:

EnbF 10 2 1 Master 10 has enabled master 2's fault

EnbH – Enable/Disable Station Hardware Fault

Register Function Code: 207

Parameters:

Master
Station
Status (0/1)

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.3b3.

Command/Response Messages:

Enables or disables faults on “Station” on the queue of “Master”. The “Status” is a 1 (one) to enable the fault and a 0 (zero) to disable the fault. If the fault is already enabled/disabled by on “Master”, then this will return a good response. If the fault can not be enabled/disabled by “Master”, then the fail response is returned. This will enable/disable the reporting of the **Halm**station faults.

Examples:

EnbH 10 1130 0 *Request to disable fault of station 1130 from master 10*

Status Message:

Reports when faults from “Master” are enabled/disabled on the DXL system (i.e., not from the host). This can be generated by the **Stat** or **AllS** command as well. Only stations which are disabled will be reported in response to the **Stat** or **AllS** command.

Examples:

EnbH 10 1130 1 *Master 10 has enabled station 1130’s fault*

Enbl – Man/Unman Master

Register Function Code: 11

Parameters[22]:

Master
State (0/1)

Implementation Status:

Command/response and status messages implemented in DXL version 0.1.0b1.

Command/Response Messages:

This command attempts to man or unman "Master". The "State" is a 1 (one) if the host computer wants to man "Master" and a 0 (zero) if the host computer wants to unman "Master". If the master can not be manned or unmanned, then this will return a response indicating that the command failed. If "Master" is already unmanned and the unman command is sent, then this will return a good response. If "Master" is already manned and the man command is sent, then this will return a good response.

Examples:

Enbl 10 0 *Unman Master 10*

Status Messages:

Reports that "Master" has been manned or unmanned. This can also report whether "Master" is manned or unmanned when the **Stat** or **AllS** command is sent. This is always sent (regardless of whether the master is unmanned or manned) when the **Stat** or **AllS** command is received.

Examples:

Enbl 10 1 *Master 10 was manned*

[22] The parameters for this command differ from the corresponding DXI command.

EnbM – Enable/Disable Station Music Button

Register Function Code: 72

Parameters[23]:

Station
State (0/1)

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.3b3.

Command/Response Messages:

Sent by the host to enable or disable the music button of "Station". "State" is a 1 (one) if the host wants to enable "Station" and a 0 (zero) if the host wants to disable the "Station". This only affects the music button operation, not the call request, tamper alarms, etc. If the music button is already enabled and a command to enable it is sent, this will return a good response. If the music button is already disabled and a command to disable it is sent, this will return a good response as well.

DXL version 3.0.1b3: If there is no station with this number then the "Station" number will be looked up to see if there is a station group which has this number. If there is a station group numbered "Station" then this command will enable the station group's members music switches as if the host sent an **EnGM**command with "Station" and "State".

Examples:

EnbM 1130 1 *Enable station 1130's music button*
EnbM 1130 0 *Disable station 1130's music button*

Status Messages:

This message reports whether the station's music button is enabled or disabled. This can also be generated when the **Stat** or **AllS** command is sent. Only stations whose music button is disabled will be reported in response to the **Stat** or **AllS** command.

Examples:

EnbM 1130 1 *Station 1130's music button is enabled*

[23] The parameters for this command differ from the corresponding DXI command.

Enbs – Enable/Disable Station Call Request

Register Function Code: 28

Parameters:

Master
Station
Status (0/1)

Implementation Status:

Command/Response and status messages implemented in DXL version 0.5.3b3.

Command/Response Messages:

Enables or disables "Station" call requests. "Master" is the master that is disabling the station. The "Status" is a 1 (one) to enable the station call requests and a 0 (zero) to disable the station call requests. If the station call request is already disabled by this master, then this will return a good response. If the station call request can not be enabled/disabled by this master, then the fail response is returned.

DXL version 3.0.1b3: If there is no station with this number then the "Station" number will be looked up to see if there is a station group which has this number. If there is a station group numbered "Station" then this command will enable the station group's members call requests as if the host sent an **EnGS**command with "Master", "Station" and "Status".

Examples:

Enbs 10 1130 0 *Request to disable station 1130's call request from master 10*

Status Message:

Reports when all call request switches from "Station" are enabled/disabled on "Master". This can be generated by the **Stat** or **AllS** command as well. All stations which are disabled when the **Stat** or **AllS** command was sent will be reported. Enabled stations are not reported.

Examples:

Enbs 10 1130 1 *Master 10 has enabled station 1130's call request*

EnbT – Enable/Disable Tamper Alarm

Register Function Code: 215

Parameters:

Master
Station
Status (0/1)

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.5

Command/Response Messages:

Enables or disables "Station" tamper alarms. "Master" is the master that is disabling the station. If "Master" is 0, then "Station" tamper alarms are disabled on all masters (at least those that can receive tamper alarms from "Station"[24]).The "Status" is a 1 (one) to enable the tamper alarm and a 0 (zero) to disable the tamper alarm. If the tamper alarm is already disabled by this master, then this will return a good response. If the tamper alarm can not be enabled/disabled by this master, then the fail response is returned. This affects all the tamper alarms from this station. This can be used to suppress the reception of **Talm**status messages.

Examples:

EnbT 10 1130 0 *Request to disable station 1130's call request from master 10*

Status Message:

Reports when all tamper alarms from "Station" are enabled/disabled on "Master". This can be generated by the **Stat** or **AllS** command as well. Only stations that have disabled tamper alarms are reported in response to the **Stat** or **AllS** command.

Examples:

EnbT 10 1130 1 *Master 10 has enabled station 1130's tamper alarm*

[24] i.e., all the masters on the secondary tree of the alarm.

EndC – End Call

Register Function Code: 10

Parameters:

Master

Implementation Status:

Command/response and status messages implemented in DXL version 0.1.0b1.

Command/Response Messages:

This command is used to end the call on the "Master". Specifying a "Master" of 0 is a syntax error. The response will be generated even if this master was not previously connected to anything (i.e., it returns a good response whether it actually does any disconnects or not). This will only end calls, not monitors.

Examples:

EndC 10 *Ends current call, if any, on master 10*
Done EndC 10 *Response to above command.*

Status Messages:

When a new connection is made without ending the previous one (i.e., suppose the user does two consecutive Icals to different stations) a status message will be generated. This only applies to end calls which are not for things which have an individual end (i.e., Iend, Mend, etc.).

Examples:

EndC 10 *Current call from master 10*

EndM – End Monitor

Register Function Code: 51

Parameters:

Master

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.3b3.

Command/Response Messages:

This command is used to end the monitoring of “Master” (regardless of what it was monitoring, be it station, visiting booth, etc.). If “Master” was not monitoring anything, then this will return a good response.

Examples:

EndM 10 *Tells master 10 to stop monitoring*

Status Messages:

Reports that “Master” is no longer monitoring (regardless of whether it was station, visiting booth, etc.).

Examples:

EndM 10 *Reports that master 10 has stopped monitoring*

EndR – End Call Recorder

Register Function Code: 98

Parameters:

Call Recorder

Implementation Status:

Command/response messages implemented in DXL version 0.5.5.
Status message unimplemented.

Command/Response Messages:

This disconnects the call recorder from whatever it is recording. The master station ID is the master which connected the call recorder (and which is now requesting to disconnect the call recorder) and the call recorder ID is the call recorder to disconnect (from the call recorder database). If the call recorder is not connected then this command will return a good response. Note that this is an alternative to calling the recorder functions (see above) with 0 as the last parameter.

Examples:

EndR 1 *Stop recording (whatever) on call recorder 1*

Status Messages:

Reports that the call recorder is no longer recording (whatever it was recording).

Examples:

EndR 1 *Call recorder 1 has stopped recording*

EndS – Stop Signal (Tone to a Zone)

Register Function Code: 25

Parameters:

Master
SIgnal

Implementation Status:

Command/response and status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

This stops “Signal” from being output. If “Signal” is 0, then all tones related to (i.e., started by) “Master” are stopped. If “Signal” is not being output on “Zone” then this will return a good response, acting the same as if “Signal” was running. If “Signal” is 0 and there were no signals being generated by “Master” this will return a good response as well. If “Master” is 0, then all tones started with “Master” equal to 0 are stopped.

Examples:

EndS 10 0 *Stop all signals which master 10 initiated.*
EndS 10 99 *Stop broadcasting signal 99*

Status Messages:

This command, when received by the host computer, indicates that the tone was ended successfully from the master station. This can be either due to the signal timing out (i.e., if it is defined to be a finite length) or some master ending the signal manually .

Examples:

EndS 10 99 *Master 10 stopped tone 5 on zone 99*

EnGM – Enable/Disable Station Group Member's Music Buttons

Register Function Code: 224

Parameters:

Station Group
State (0/1)

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.3b3.

Command/Response Messages:

Sent by the host to enable or disable the music button of all members of "Station Group". "State" is a 1 (one) if the host wants to enable "Station Group" members' music buttons and a 0 (zero) if the host wants to disable the "Station Group" members' music buttons. This only affects the music button operation, not the call request, tamper alarms, etc. If the music button is already enabled and a command to enable it is sent, this will return a good response. If the music button is already disabled and a command to disable it is sent, this will return a good response as well.

Examples:

EnGM 1130 1 *Enable members of station group 1130's music buttons*
EnGM 1130 0 *Disable members of station group 1130's music buttons*

Status Messages:

This message reports whether the station's music button is enabled or disabled. This can also be generated when the **Stat** or **AllS** command is sent. Only stations whose music button is disabled will be reported in response to the **Stat** or **AllS** command.

Examples:

EnGM 1130 1 *Members of station group 1130's music buttons were enabled*

EnGS – Enable/Disable Station Group Member's Call Requests

Register Function Code: 222

Parameters:

Master
Station Group
Status (0/1)

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.3b3.

Command/Response Messages:

Enables or disables all "Station Group" member's call requests. "Master" is the master that is disabling the station group. The "Status" is a 1 (one) to enable the station call requests and a 0 (zero) to disable the station call requests. If the station call request is already disabled by this master, then this will return a good response. If the station call request can not be enabled/disabled by this master, then the fail response is returned.

Examples:

EnGS 10 1130 0 *Request to disable members of station group 1130's call requests from master 10*

Status Message:

Reports when all call request switches from "Station Group" are disabled. This can be generated by the **Stat** or **AllS** command as well. All stations which are disabled when the **Stat** or **AllS** command was sent will be reported. Enabled station groups are not reported.

Examples:

EnGS 10 1130 1 *Master 10 has enabled members of group 1130's call requests*

EnGT – Enable/Disable Station Group Member's Tamper Alarms

Register Function Code: 225

Parameters:

Master
Station Group
Status (0/1)

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.5.

Command/Response Messages:

Enables or disables all members of "Station Group" tamper alarms. "Master" is the master that is disabling the station. If "Master" is 0, then "Station Group" tamper alarms are disabled on all masters (at least those that can receive tamper alarms from members of "Station Group"[25]). The "Status" is a 1 (one) to enable the tamper alarm and a 0 (zero) to disable the tamper alarm. If the tamper alarm is already disabled by this master, then this will return a good response. If the tamper alarm can not be enabled/disabled by this master, then the fail response is returned. This affects all the tamper alarms from this station.

Examples:

EnGT 10 1130 0 *Request to disable members of station group1130's call requests from master 10*

Status Message:

Reports when all tamper alarms from "Station" are disabled. This can be generated by the **Stat** or **AllS** command as well. Only stations that have disabled tamper alarms are reported in response to the **Stat** or **AllS** command.

Examples:

EnGT 10 1130 1 *Master 10 has enabled members of station group 1130's tamper alarm*

[25] i.e., all the masters on the secondary tree of the alarm.

Fail – Failure Status

Register Function Code: 17

Parameters:

Function Code and Parameters.

Implementation Status:

Response message implemented in DXL version 0.1.0b1.

Response Message:

This response message indicates that the command failed to be executed. The parameter string is an echo of the command which caused the problem. This can result from hardware faults etc.

Examples:

Fail Ical 10 1130 *Command "Ical 10 1130" failed.*

Status Messages:

N/A.

GLev – Set Audio Level Alarm Level of Station Group Members

Register Function Code: 227

Parameters:

Station Group
Level (0-4)

Implementation Status:

Command/response messages implemented in DXL version 0.5.5.
Status message implemented in DXL version 2.1.0.

Command/Response Messages:

This command changes the audio level alarm level of all members of "Station Group". A "Level" of 0 means that the alarm is disabled (if already disabled then this command will return a good response). Turning off the alarm does not cancel any existing alarms from members of "Station Group". If "Levl" is between 1 and 4 then the alarm is set to one of these levels (if already at the appropriate level then this command will return a good response).

Examples:

GLev 1130 0 *Turn off alarm monitoring on members of station group 1130*
GLev 1130 2 *Set level alarm of members of station group 1130 to second set of parameters*

Status Messages:

Reports that the audio level alarms have been changed (either to levels 1-4 or disabled with 0). This can be sent in response to a **Stat** or **AllS** command. Only those alarms whose level is not 0 will be reported in response to the **Stat** or **AllS** command.

Examples:

GLev 1130 1 *Station group 1130's members audio level changed*

GMas – Copy Master Music Selection To Station Group

Register Function Code: 229

Parameters:

Master
Station Group

Implementation Status:

Command/response messages implemented in DXL version 2.1.0.

Command/Response Messages:

This command is sent to change the music on members of “Station Group” to the same channel as what is currently on “Master”. If there is no music on “Master” then there will be no music placed on the “Station Group” (if it was connected to a music channel, it will be disconnected). If there is music on “Master” then it will be placed on “Station Group” (this can result in the music on members of “Station Group” being changed). If the music on “Station Group” was the same as on “Master” then this will return a good response. If there is no music on “Master” and no music on the “Station Group” then this will return a good response as well.

Examples:

GMas 10 1130 *Change members of station group 1130’s music channel to the same as master 10*

Status Messages:

N/A.

GMus – Station Group Music Selection

Register Function Code: 228

Parameters[26]:

Station Group
Music
Low/High Volume

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.3b3.

Command/Response Messages:

Sent from the host to tell all members "Station Group" to change their music channel to "Music". This can be done even when some members of "Station Group" are in a call (it is assumed that music is lower priority than calls so the actual music will not be present until the call is ended). When "Music" is 0, then the music is disconnected from the members of "Station Group" (the "Low/High Volume" should be zero in this case). This returns a good response if "Music" is the same channel that is currently connected. The "Low/High Volume" is used to select the appropriate volume level of the music channel, zero meaning select the low volume and non-zero meaning select the high volume.

Examples:

GMus 1130 2 0 *Change station group 1130' s members to music 2, low*
Gmus 1130 20 1 *Change station group1130's members to music 20, high*

Status Messages:

N/A.

[26] The parameters for this command differ from the corresponding DXI command.

Hack – Station Fault Acknowledge

Register Function Code: 206

Parameters:

Master
Station

Implementation Status:

Command/response and status messages implemented in DXL version 0.1.0b1.

Command/Response Messages:

This is sent to acknowledge a hardware alarm on "Station" by "Master" (i.e., to move it from active to acknowledged queue). If the alarm is already on the acknowledged queue, then this will return a good response. If the alarm is not active or already acknowledged then this will return a fail response.

Examples:

Hack 10 1130 *Acknowledge fault on Station 1130 on master 10*

Status Messages:

Reports that "Station" hardware alarm has been acknowledged by "Master" (i.e., placed on its acknowledged queue). This can be generated due to a **Stat** or **AllS** command on the master. Only stations which are acknowledged will be generated in response to the **Stat** or **AllS** command.

Examples:

Hack 10 1130 *Fault detected on Station 1130 was acknowledged on master 10*

Halm – Station Fault

Register Function Code: 19

Parameters:

Master
Station
Status (0/1)

Implementation Status:

Status message implemented in DXL version 0.1.0b1. Changed from register function code 18 to 19 to be compatible with DXI in DXL version 0.5.3B1.

Command/Response Messages:

N/A.

Status Messages:

This status message is sent when a hardware alarm from "Station" is queued at "Master" or when it is cancelled from "Master". If "Status" is set to 1 (one), then the fault has been detected and queued on "Master". If "Status" is a 0 (zero), then the fault has been cancelled from "Master" (typically due to the source of the fault(s) being cleared). This can be generated due to a **Stat** or **AllS** command on the master. Only stations which have active, unacknowledged faults are reported in response to the **Stat** or **AllS** command. These can be enabled/disabled via the **EnbH**host command.

Examples:

Halm 10 1130 1 *Hardware fault detected on Station 1130 is queued on master 10*
Halm 10 1130 0 *Hardware fault cleared on Station 1130 removed from master 10*

Ical – Station Call

Register Function Code: 7

Parameters:

Master
Station

Implementation Status:

Command/response and status messages implemented in DXL version 0.1.0b1.

Command/Response Messages:

Attempts a call from "Master" to "Station". Specifying a "Station" of 0 will disconnect the current call (see **Iend/Mend/Pend**). If "Master" is in a call to something else (i.e., a master, station, etc.) then this will disconnect that call (and generate an **Iend/Mend/Pend** for "Master") before trying to connect to "Station". The **Iend/Mend/Pend**, in this case, will always be sent before the response. If "Master" was doing a monitor, then that will be disconnected (and generate an **EndM** for "Master") before trying to connect to "Station". The **EndM**, in this case, will always be sent before the response. Specifying "Master" as 0 is a syntax error. The good response will be generated if the connection is made or the device to which "Master" was previously calling was "Station". In the second case, the disconnection is not done and "Master" and "Station" are left connected. If the "Station" is busy in a higher priority call, then this command will generate a busy response.

Examples:

Ical 10 1130 *Call station 1130 from master 10.*

Status Messages:

Reports that "Station" has been connected to "Master". This can also report what station (if any) "Master" is connected to when the **Stat** or **AllS** command is sent. If no station is connected to this master, then the message is not generated (i.e., is not sent with a station of 0).

Examples:

Ical 10 1130 S*tation 1130 was called from master 10.*

Ican – Station Call Request Cancel

Register Function Code: 2

Parameters:

Master
Station

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.3b3.

Command/Response Messages:

Cancels the call request from “Station” at “Master”. Sends a good response even if “Station” did not have a call request on “Master”. Specifying “Master” or “Station” as 0 is a syntax error. If the call request is not present (i.e., not active or acknowledged) then this will return a good response (as if it was actually cancelled).

Examples:

Ican 10 1130 *Cancel call request from station 1130 to master 10.*

Status Messages:

Reports that a call request from “Station” was canceled at “Master”.

Examples:

Ican 10 1130 *Call request from station 1130 cancelled at master 10.*

Icrq – Station Call Request

Register Function Code: 1

Parameters:

Master
Station

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.3b3.

Command/Response Messages:

Causes a station call request alarm from "Station" to be queued at "Master". The "Master" that this call request is queued is specified as "Master". This always responds with a good response even though it may (if the station's call request is disabled) not queue the call request at the "Master" but instead may queue it at the secondary master of "Master". If this command is successful, then the status message is generated to indicate that the command has been processed (if the call request is not disabled, see **Enbs**).

Examples:

Icrq 1 1130 *Call request from station 1130 will be queued at master 1.*
Done Icrq 1 *1130 Call request from station 1130 may be queued at master 1.*

Status Messages:

Reports that a call request from "Station" was queued at "Master". This can also report the station call requests queued on "Master" when the **Stat** or **AllS** command is sent. All station call requests on "Master" will be reported with this status message.

Examples:

Icrq 10 1130 *Call request from station 1130 queued at master 10.*

Iend – End Call To Station

Register Function Code: 200

Parameters:

Master
Station

Implementation Status:

Command/response and status messages implemented in DXL version 0.1.0b1.

Command/Response Messages:

This command is used to end the call on the "Master". Specifying a "Master" of 0 is a syntax error. If "Station" is 0, then any existing station call will be ended (if there was a page or master call, then this would not be disconnected and this would return a failure). If "Station" is not 0, then "Master" will only be disconnected if it is connected to "Station". The response will be generated even if this master was not previously connected to anything (i.e., it returns a good response whether it actually does any disconnects or not). This will only end calls, not monitors.

Examples:

Iend 10 0 *Ends current call, if any, on master 10*
Done Iend 10 1130 *Response to above command if connected to station 1130*
Iend 10 1130 *Ends current call on master 10 but only if from station 1130*
Done Iend 10 1130 *Response to above command if connected to station 1130*

Status Messages:

When a new connection is made without ending the previous one (i.e., suppose the user does two consecutive Icals to different stations) a status message will be generated. This will be generated to tell the host which station call has been ended. This will not be generated if the call ended is not to a station. If there is an EndC used to end the call this will not be generated.

Examples:

Iend 10 1130 *Current call from master 10 to station 1130 was ended*

Imon – Monitor Station From Master

Register Function Code: 48

Parameters:

Master
Station

Implementation Status:

Command/response and status messages implemented in DXL version 0.1.0b1.

Command/Response Messages:

Sent to tell "Master" to monitor "Station". If "Station" is 0, then "Master" will stop monitoring whatever it was previously (if anything) monitoring, regardless of whether it was a station or not. This is the same as the **EndM** command. If "Master" was not monitoring anything, then this will result in a good response. If "Station" is not 0, then this "Master" will monitor "Station". If "Master" was already monitoring "Station" then this will result in a good response but the connection will be left. If "Station" is not 0 and this master was monitoring some other device, this device will be disconnected (which will result in an **EndM** status message being sent before the **Imon** response is generated).

Examples:

Imon 10 1130 *Monitor station 1130 on master 10*
Imon 10 0 *Stop monitoring on master 10*

Status Messages:

Reports that "Master" is now monitoring "Station". This can also be sent in response to a **Stat** or **AllS** command (if a monitor was in progress). This is only reported (in response to the **Stat** or **AllS** command) if the station being monitored is not 0.

Examples:

Imon 10 1130 *Master 10 is now monitoring station 10*

IRec – Record Station

Register Function Code: 93

Parameters[27]:

Call Recorder
Station

Implementation Status:

Command/response messages implemented in DXL version 0.5.5.
Status message unimplemented.

Command/Response Messages:

Connects "Station" to the "Call Recorder". "Station" is the station that is being recorded. "Call Recorder" can not be 0. If the "Station" is 0, then the call recorder is disconnected (from whatever it was connected to, even if that was not a station, exactly like doing an **EndR**). If "Station" was already connected to the "Call Recorder", then this will return a good response (if "Call Recorder" is not 0). If "Station" is not connected (and "Call Recorder" is 0) then this will also return a good response. If "Call Recorder" was connected to any other device, then this will be disconnected and an **EndR** command will be generated before this command's response is received by the host.

Examples:

IRec 1 1130 *Record station10 on call recorder 1*

Status Messages:

Reports that "Station" is being recorded by "Call Recorder". This can be also be sent in response to a **Stat** or **AllS** command. Only call recorders which are recording stations will be sent this messages in response to the **Stat** or **AllS** command.

Examples:

IRec 1 10 *Station 10 is being recorded by call recorder 1*

[27] The parameters for this command differ from the corresponding DXI command.

Iset – Start Station to Station Call

Register Function Code: 103

Parameters[28]:

Station
Station

Implementation Status:

Command/response and status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

Attempts a call from the first "Station" to the second "Station". If either "Station" is in a call to something else (i.e., a master, station, etc.) then this will disconnect that call (and generate an **Istp** for either or both "Station") before trying to connect to the other. The **Istp**, in this case, will always be sent before the response. Specifying either "Station" as 0 is an error. The good response will be generated if the connection is made or if connection had previously been made. In the second case, the disconnection is not done and the first "Station" and second "Station" are left connected. If either the first or second "Station" is busy in a higher priority call, then this command will generate a busy response.

Examples:

Iset 1101 1130 *Call station 1130 from 1101*

Status Messages:

Reports that the first "Station" has been connected to the second "Station". This can also be generated by the **Stat** or **AllS** command. Only stations involved in a call are reported in response to the **Stat** or **AllS** command.

Examples:

Iset 1101 1130 S*tation 1101 and 1130 were connected*

[28] The parameters for this command differ from the corresponding DXI command.

Istp – End Station To Station Call

Register Function Code: 104

Parameters[29]:

Station
Station

Implementation Status:

Command/response and status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

This command is used to end the between the two "Stations". If neither "Station" is connected to anything, then this will return a good response. If the "Stations" were connected together then they will be disconnected and a good response returned. If either "Station" is involved in another call, this will generate a fail response. Specifying either "Station" as 0 is an error.

Examples:

Istp 1101 1130 *End station 1101 to 1130 connection*

Status Messages:

Reports when station to station call (not visiting booth) is ended.

Examples:

Istp 1101 1130 *Connection between stations 1101 1130 has ended.*

[29] The parameters for this command differ from the corresponding DXI command.

IVad – Station Volume Adjustment

Register Function Code: 22

Parameters:

Station
Increment (0/1)

Implementation Status:

Command/response messages implemented in DXL version 0.1.0b1.

Command/Response Messages:

This command changes the volume of “Station”. If “Increment” is 0 (zero), then the station’s volume will decrement by 1 setting (i.e., get quieter). If “Increment” is 1 (one), then the station’s volume will increment by 1 setting (i.e., get louder). Note that the good response is always sent back, regardless of whether or not the volume can increase or decrease any more than its current setting (i.e., whether it is at its maximum or minimum respectively).

Examples:

IVad 1130 1 *Increment station 1130’s volume by 1*
1
IVad 1130 0 *Decrement station 1130’s volume by 1*

Status Messages:

N/A.

Levl – Set Audio Level Alarm Level

Register Function Code: 47

Parameters:

Station
Level (0-4)

Implementation Status:

Command/response messages implemented in DXL version 0.5.5.
Status message implemented in DXL version 2.1.0.

Command/Response Messages:

This command changes the audio level alarm level of "Station". A "Level" of 0 means that the alarm is disabled (if already disabled then this command will return a good response). Turning off the alarm does not cancel any existing alarms from "Station". If "Levl" is between 1 and 4 then the alarm is set to one of these levels (if already at the appropriate level then this command will return a good response).

DXL version 3.0.1b3: If there is no station with this number then the "Station" number will be looked up to see if there is a station group which has this number. If there is a station group numbered "Station" then this command will change the station group's members audio level alarm levels as if the host sent a **GLev**command with "Station" and "Level".

Examples:

Levl 1130 0 *Turn off alarm monitoring on station 1130*
Levl 1130 2 *Set level alarm of station 1130 to second set of parameters*

Status Messages:

Reports that the audio level alarms have been changed (either to levels 1-4 or disabled with 0). This can be sent in response to a **Stat** or **AllS** command. Only those alarms whose level is not 0 will be reported in response to the **Stat** or **AllS** command.

Examples:

Levl 1130 1 *Station 1130's audio level was changed to first set of parameters*

Mack – Master Hardware Fault Acknowledge

Register Function Code: 220

Parameters:

Master
Master

Implementation Status:

Command/response messages implemented in DXL version 0.5.3b3.
Status message implemented in DXL version 0.1.0b1.

Command/Response Messages:

Sent to acknowledge the second “Master” hardware alarm on the first “Master” alarm queue. If the alarm has already been acknowledged (and placed on the first “Master” alarm queue) then this will return a good response. If the alarm is not active or has already been acknowledged, then this will return a fail response.

Examples:

Mack 10 1130 *Fault on Master 1130 acknowledged (queued on master 10)*

Status Messages:

Sent when a hardware alarm from the second “Master” (which is queued at the first “Master”) is acknowledged. This can be generated due to a **Stat** or **AllS** command on the master. Only masters that have acknowledged faults are reported in response to the **Stat** or **AllS** command.

Examples:

Mack 10 1130 *Fault on Master 1130 acknowledged (queued on master 10)*

Mcal – Master Call

Register Function Code: 8

Parameters:

Master
Master

Implementation Status:

Command/response and status messages implemented in DXL version 0.1.0b1.

Command/Response Messages:

Attempts a call from the first "Master" to the second "Master". Specifying a second "Master" of 0 will disconnect the current call (see **Iend/Mend/Pend**). If the first "Master" is in a call to something else (i.e., a master, station, etc.) then this will disconnect that call (and generate an **Iend/Mend/Pend** for the first "Master") before trying to connect to the second "Master". The **Iend/Mend/Pend**, in this case, will always be sent before the response. If the device to which the first "Master" was calling was second "Master" then the response is sent and the disconnection is not done. If the first "Master" was doing a monitor, then that will be disconnected (and generate an **EndM** for the first "Master") before trying to connect to the second "Master". The **EndM**, in this case, will always be sent before the response. Specifying the first "Master" as 0 is a syntax error. The good response will be generated if the connection is made or the device to which the first "Master" was previously calling was the second "Master". In the second case, the disconnection is not done and two "Master" are left connected. If the second "Master" is busy in a higher priority call, then this command will generate a busy response.

Examples:

Mcal 10 14 *Call master 14 from master 10.*

Status Messages:

Reports that first "Master" has been connected to the second "Master". This can also report what master (if any) "Master" is connected to when the **Stat** or **AllS** command is sent. If no master is connected to this master, then the message is not generated (i.e., is not sent with a master of 0)

Examples:

Mcal 10 14 *Master 14 was called from master 10.*

Mcan – Master Call Request Cancel

Register Function Code: 4

Parameters:

Master
Master

Implementation Status:

Status message implemented in DXL version 0.1.0b1.

Status Messages:

This message reports that a call request from the second “Master” was canceled at the first “Master”.

Examples:

Mcan 10 1130 C*all request from master 1130 cancelled by master 10.*

Mcrq – Master Call Request

Register Function Code: 3

Parameters:

Master
Master

Implementation Status:

Command/response and status messages implemented in DXL version 0.1.0b1.

Command/Response Messages:

This message reports that a master call request from the second “Master” was received and queued at the first “Master”. Specifying either “Master” as 0 is a syntax error. This never sends a good response to the host. If this command is successful, then the status command is generated to indicate that the command has been processed.

Examples:

Mcrq 10 1130 *Call request from master 1130 will be queued at master 10.*

Status Messages:

This message reports that a call request from the second “Master” was queued at the first “Master”. This can also report the master call requests queued on “Master” when the **Stat** or **AllS** command is sent. All master call requests on “Master” will be reported with this status message.

Examples:

Mcrq 10 1130 *Call request from master 1130 queued by master 10.*

MDdn – Decrease Master Auto-Monitor Rate

Register Function Code: 115

Parameters:

Master
Seconds (1-3600)

Implementation Status:

Command/response messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

This command is sent to decrease the master's monitor rate by a specific amount. The monitor rate can be between 1 and 3600 seconds. This time is the amount of time that each station's monitor lasts for, not the total time taken to go through all the members in the master's monitor list. For example, if the current rate was every 30 seconds, sending a parameter of 20 would cause the monitoring to occur every 10 seconds. The decrease will not decrease the monitor rate beyond 1 second. If the decreased setting would be less than 1 second, then this will return a Fail response otherwise it will return a Done response. This will always return a "Done" even if the increase would have made the monitoring time less than 1 second.

Examples:

MDdn 1 99 *Increase master's monitor rate by 99 seconds*

Status Messages:

N/A.

MDof – Background Monitoring of Station Stopped

Register Function Code: 234

Parameters:

Master
Station

Implementation Status:

Status message implemented in DXL version 1.1.0b4.

Status Messages:

Reports that “Master” has stopped performing a background monitoring of “Station” at this particular time.

Examples:

MDof 1 10 *Master 1 has stopped monitoring station 10*

Status Messages:

N/A.

MDon – Background Monitoring of Station Started

Register Function Code: 233

Parameters:

Master
Station

Implementation Status:

Status message implemented in DXL version 1.1.0b4.

Status Messages:

Reports that “Master” has started performing a background monitoring of “Station” at this particular time.

Examples:

MDof 1 10 *Master 1 has started monitoring station 10*

MDst – Set Master Auto-Monitor Rate

Register Function Code: 113

Parameters:

Master
Seconds (1-3600)

Implementation Status:

Command/response messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

This command is sent to change the master's monitor rate to a specific value. The monitor rate can be between 1 and 3600 seconds. This time is the amount of time that each station's monitor lasts for, not the total time taken to go through all the members in the master's monitor list.

Examples:

MDst 1 99 *Set master's monitor rate to 99 seconds*

Status Messages:

N/A.

MDup – Increase Master Auto-Monitor Rate

Register Function Code: 114

Parameters:

Master
Seconds (1-3600)

Implementation Status:

Command/response messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

This command is sent to increase the master's monitor rate by a specific amount. The monitor rate can be between 1 and 3600 seconds. This time is the amount of time that each station's monitor lasts for, not the total time taken to go through all the members in the master's monitor list. . For example, if the current rate was every 10 seconds, sending a parameter of 20 would cause the monitoring to occur every 30 seconds. The increase will not increase the monitor rate beyond 3600 seconds. This will always return a "Done" even if the increase would have made the monitoring time greater than 3600 seconds.

Examples:

MDup 1 99 *Increase master's monitor rate by 99 seconds*

Status Messages:

N/A.

Mend – End Call To Master

Register Function Code: 201

Parameters:

Master
Master

Implementation Status:

Command/response messages implemented in DXL version 0.1.0b1.
Status message implemented in DXL version 0.5.3b3.

Command/Response Messages:

This command is used to end the call on the first “Master”. Specifying a first “Master” of 0 is a syntax error. If the second “Master” is 0, then any existing master call will be ended. If the second “Master” is not 0, then the first “Master” will only be disconnected if it is connected to the second “Master” or if it was not connected to anything. The response will be generated even if this master was not previously connected to anything (i.e., it returns a good response whether it actually does any disconnects or not). This will only end calls, not monitors. When this is called and a call is ended then a status message will be reported for the second “Master” (i.e., Mend for second “Master” will occur at roughly the same time).

Examples:

Mend 10 0 *Ends current page, if any, on master 10*
Done Mend 10 14 *Response to above Mend command if master 14 was connected.*
Mend 10 14 *Ends current page on master 10 but only if from master 14*
Done Mend 10 14 *Response to above Men command if master 14 was connected.*

Status Messages:

When a new connection is made without ending the previous one (i.e., suppose the user does two consecutive Mcals to different masters) a status message will be generated. This will be generated to tell the host which master call has been ended. This will not be generated if the call ended is not a master call. If there is an EndC used to end the call this will not be generated.

Examples:

Mend 10 1130 *Current call from master 10 to master 1130 was ended*

Mflt – Master Hardware Fault

Register Function Code: 219

Parameters:

Master
Master
Status (0/1)

Implementation Status:

Status message implemented in DXL version 0.1.0b1.

Command/Response Messages:

N/A.

Status Messages:

Sent when a hardware alarm from the second “Master” is queued at the first “Master” or when it is cancelled from the first “Master”. If “Status” is set to 1 (one), then the fault has been detected and queued on the first “Master”. If “Status” is a 0 (zero), then the fault has been cancelled from the first “Master” (typically due to the source of the fault(s) being cleared). This can be generated due to a **Stat** or **AllS** command on the master. Only masters that have active, unacknowledged faults are reported in response to the **Stat** or **AllS** command. These can be enabled/disabled with the **EnbF** host command.

Examples:

Mflt 10 1130 1 *Hardware fault detected on Master 1130 is queued on master 10*
Mflt 10 1130 0 *Hardware fault cleared on Master 1130 removed from master 10*

Mmut – Master Microphone Mute Override

Register Function Code: 250

Parameters[30]:

Master
Mic Mute State

Implementation Status:

Command/response and messages implemented in DXL version 2.3.0.

Command/Response Messages:

This is used to implement a master microphone mute from a host. When the "Mic Mute State" is set to 1 then the "Master" audio will function as if its MM switch had been pressed and allow the master to listen to the its connected device (assuming the master is controlling the conversation). When this is set to 0 then the master's MM will act as if the "Master" MM is not pressed and allow the master's VOX to control the direction of the conversation. Note that setting this to a 1 will override any hardware MM switches wired to the master. When this is set to 0 then the hardware MM switches can control the audio direction.

Examples:

Mmut 100 1 *Override master 100's MM switch to allow master to listen.*
Mmut 100 0 *Turn off master MM, allowing VOX to control the conversation.*

Status Messages:

Reports that "Master" has its microphone mute override (via the host) set. This can be also be sent in response to a **Stat** or **AllS** command. Only masters which have their microphone mute set via the host will have this messages sent in response to the **Stat** or **AllS** command. *This does not report the current state of the hardware microphone-mute switch.*

Examples:

Mmut 10 1 *Master 10's microphone mute override is currently set.*

[30] The parameters for this command differ from the corresponding DXI command.

Mptt – Master PTT Override

Register Function Code: 249

Parameters[31]:

Master
PTT State

Implementation Status:

Command/response and status messages implemented in DXL version 2.3.0.

Command/Response Messages:

This is used to implement a master PTT from a host. When the "PTT State" is set to 1 then the "Master" audio will function as if its PTT switch had been pressed and allow the master to talk to the its connected device (assuming the master is controlling the conversation). When this is set to 0 then the master's PTT will act as if the "Master" PTT is not pressed and allow the connected device to talk to the master. Note that setting this to a 1 will override any hardware PTT switches wired to the master. When this is set to 0 then the hardware PTT switches can control the audio direction.

Examples:

Mptt 100 1 *Override master 100's PTT switch to allow master to talk.*
Mptt 100 0 *Turn off master PTT, allowing connected device to talk to master 100.*

Status Messages:

Reports that "Master" has its push-to-talk override (via the host) set. This can be also be sent in response to a **Stat** or **AllS** command. Only masters which have their push-to-talk set via the host will have this messages sent in response to the **Stat** or **AllS** command. *This does not report the current state of the hardware push-to-talk switch.*

Examples:

Mptt 10 1 *Master 10's push-to-talk override is currently set.*

[31] The parameters for this command differ from the corresponding DXI command.

MRec – Record Master

Register Function Code: 97

Parameters[32]:

Call Recorder
Master

Implementation Status:

Command/response messages implemented in DXL version 0.5.5.
Status message unimplemented.

Command/Response Messages:

Connects "Master" to the "Call Recorder". "Master" is the master that is being recorded. "Call Recorder" can not be 0. If the "Master" is 0, then the call recorder is disconnected (from whatever it was connected to, even if that was not a master, exactly like doing an **EndR**)[33]. If "Master" was already connected to the "Call Recorder", then this will return a good response (if "Call Recorder" is not 0). If "Master" is not connected (and "Call Recorder" is 0) then this will also return a good response. If "Call Recorder" was connected to any other device, then this will be disconnected and an **EndR** command will be generated before this command's response is received by the host.

Examples:

MRec 1 10 *Record master10 on call recorder 1*

Status Messages:

Reports that "Master" is being recorded by "Call Recorder". This can be also be sent in response to a **Stat** or **AllS** command. Only call recorders which are recording stations will be sent this messages in response to the **Stat** or **AllS** command.

Examples:

MRec 1 10 *Master 10 is being recorded by call recorder 1*

[32] The parameters for this command differ from the corresponding DXI command.
[33] Note that this does not apply to the master call recorder but to the call recorder database. Master call recorders are dedicated to masters and can never be disconnected (except by changing the configuration). Call recorders are dynamic and can be connected/disconnected from (almost) any device in the system.

MusM – Master Music Selection

Register Function Code: 59

Parameter:

Master
Music
Low/High Volume

Implementation Status:

Command/response messages implemented in DXL version 0.5.3b3.
Status message implemented in DXL version 2.0.1b3.
Syntax changed for version 0.5.5B5 to include volume level for master music.

Command/Response Messages:

Sent from the host to tell "Master" to change its music channel to "Music". This can be done even when "Master" is in a call or monitor (it is assumed that music is lower priority than calls so the actual music will not be present until the master ends the current call or monitor). When "Music" is 0, then the music is disconnected from "Master" (the "Low/High Volume" should be zero in this case). This returns a good response if "Music" is the same channel that is currently connected. The "Low/High Volume" is used to select the appropriate volume level of the music channel, zero meaning select the low volume and non-zero meaning select the high volume.

Examples:

MusM 10 2 *Switch master 10 to music channel 2*

Status Messages:

This will be sent in response to a **Stat** or **AllS** command. Only masters which have music are reported in response to the **Stat** or **AllS** command.

Examples:

MusM 10 1 *Music channel of master 10 was changed to music channel 1*
MusM 10 0 *Music channel of master 10 was disconnected*

MusN – Next Master Music Selection

Register Function Code: 60

Parameters:

Master

Implementation Status:

Command/response messages implemented in DXL version 0.5.3b3.

Command/Response Messages:

Sent from the host to tell "Master" to change its music channel to the next music channel. Once all channels have been cycled through then this will cause the music to be disconnected. This can be done even when the master is in a call or monitor (it is assumed that music is lower priority than calls so the actual music will not be present until the master ends the current call or monitor). This returns a good response as long as there are music channels defined in the system.

Examples:

MusN 10 2 *Change master 10 to next music channel (or disconnect)*

Status Messages:

N/A.

MusP – Previous Master Music Selection

Register Function Code: 61

Parameters:

Master

Implementation Status:

Command/response messages implemented in DXL version 0.5.3b3.

Command/Response Messages:

Sent from the host to tell "Master" to change its music channel to the previous music channel. Once all channels have been cycled through then this will cause the music to be disconnected. This can be done even when the master is in a call or monitor (it is assumed that music is lower priority than calls so the actual music will not be present until the master ends the current call or monitor). This returns a good response as long as there are music channels defined in the system.

Examples:

MusP 10 2 *Change master 10 to next music channel (or disconnect)*

Status Messages:

N/A.

MVad – Master Volume Adjustment

Register Function Code: 23

Parameters:

Master
Increment (0/1)
Destination (0/1)

Implementation Status:

Command/response messages implemented in DXL version 0.1.0b1.

Command/Response Messages:

This command changes the volume of "Master". If "Increment" is 0 (zero), then the master's volume will decrement by 1 setting (i.e., get quieter). If "Increment" is 1, then the master's volume will increment by 1 (i.e., get louder). This can be sent if "Master" is in a call or not. Note that the good response is always sent back, regardless of whether or not the volume can increase or decrease any more than its current setting(i.e., whether it is at its maximum or minimum respectively). The fail response only indicates an error in the syntax of this command.

The third parameter is used to determine whether or not the volume being set is the master's volume (i.e., the volume at the master's speaker) or the destination volume (i.e., whatever is connected to the master). This can be used to set whatever is connected to the master to a new volume (though only if it is a station or zone). If this parameter is 0 (zero) then the master's speaker volume will be affected. If this parameter is 1 (one) then this will affect whatever the master is connected to's volume. The destination is specified and there is no connection currently in progress then this has no effect though it will still return a good return code.

Examples:

MVad 10 1 0 *Increment master 10's volume*
MVad 10 0 1 *Decrement whatever master 10 is connected to's volume.*

Status Messages:

N/A.

Next – Acknowledge Next Call Request

Register Function Code: 21

Parameters:

Master

Implementation Status:

Command implemented in DXL version DXL version 0.4.0B5.

Command/Response Messages:

Command to acknowledge the next call request on “Master” queue. The call request can be either a master or station call request. If there are no call requests queued up, then this will return a fail response. If there is a station call request, then the intercom will return a **Ican** to cancel the alarm (see above) and then a **Ical** to indicate which station has been called. Similarly, if there is a master call request, then the intercom will return a **Mcan** and then a **Mcal**. This will return a good response if there is at least one call request to be acknowledged. The response will occur before the **Ican** or **Mcan**, which ever is applicable, is sent. If this is sent to a master which has a call in progress the call will be ended and this will return a good response.

Examples:

Next 10 *Acknowledge next call request on master 10*

Status Messages:

N/A.

NOOP – No Operation

Register Function Code: 30

Parameters:

String (ASCII protocols only)
Numbers (Register protocols only)

Implementation Status:

Command/response and status messages implemented in DXL version 0.1.0b1.

Command/Response Messages:

When the DXL receives this command, it echoes back the entire NOOP command (with the appropriate prefix). ASCII based protocols can be passed any string (does not have to have numeric parameters). “String” is limited to a maximum of 30 characters (maximum message length is 40 characters and the response will consume up to 10 characters, i.e., “Done NOOP “). “Numbers” is limited to the number of registers in the command.

Examples:

NOOP Hello world *Causes DXL to respond with “NOOP Hello world”*

Status Messages:

The DXL does not expect the host to respond to the NOOP command. This is only generated by the DXL to prevent inactivity timeouts on the host. There is no specification about what the DXL will generate for the string/parameters of the NOOP command (can be anything or nothing[34]).

Examples:

NOOP 1 *NOOP command generated to prevent inactivity*

[34] Typically it is a counter of the number of NOOP commands sent (which must, obviously, wrap around to 0) but this is not required.

Page – Page To Zone

Register Function Code: 9

Parameters:

Master
Zone

Implementation Status:

Command/response and status messages implemented in DXL version 0.1.0b1.

Command/Response Messages:

Attempts a page from “Master” to “Zone”. Specifying a “Zone” of 0 will disconnect the current call (see **Iend/Mend/Pend**). If “Master” is in a call to something else (i.e., a master, station, etc.) then this will disconnect that call (and generate an **Iend/Mend/Pend** for “Master”) before trying to connect to “Zone”. The **Iend/Mend/Pend**, in this case, will always be sent before the response. If “Master” was doing a monitor, then that will be disconnected (and generate an **EndM** for “Master”) before trying to connect to “Zone”. The **EndM**, in this case, will always be sent before the response. Specifying “Master” as 0 is a syntax error. The good response will be generated if the connection is made or the device to which “Master” was previously calling was “Zone”. In the second case, the disconnection is not done and “Master” and “Zone” are left connected. If the “Zone” is busy in a higher priority call, then this command will generate a busy response.

Examples:

Page 10 99 *Page zone 99 from master 10*

Status Messages:

Reports that “Zone” has been connected to “Master”. This can also report what zone (if any) “Master” is connected to when the **Stat** or **AllS** command is sent. If no zone is connected to this master, then the message is not generated (i.e., is not sent with a zone of 0)

Examples:

Page 10 99 *Master 10 paged zone 99*

PAlm – Primary Alarm

Register Function Code: 57

Parameters:

Secondary Master
Master
State (0/1)

Implementation Status:

Status message unimplemented.
The syntax of this message is subject to change in future releases.

Command/Response Messages:

N/A.

Status Messages:

Sent to the host to indicate "Secondary Master" received a secondary alarm (i.e., a timeout) from primary "Master. This is treated as an alarm on "Secondary Master" This is only sent if the DXL is configured to send secondary alarms on a per master basis (i.e., rather than on per alarm basis). This is sent with a 'state' of 1 (one) to indicate that there are alarms and 0 (zero) to indicate that the alarm has been cleared. This can also be sent in response to a **Stat** or **AllS** command. Only masters which have primary alarms are reported in response to the **Stat** or **AllS** command.

Examples:

PAlm 10 1 1 *Primary master 1's timeout alarm has been queued on master 10*
PAlm 10 1 0 P*rimary master 1's timeout has been cleared on master 10*

Pcan – Panic Alarm Call Request Cancel

Register Function Code: 248

Parameters:

Master
Station

Implementation Status:

Command/response and status messages implemented in DXL version 2.3.0.

Command/Response Messages:

Cancels the panic alarm call request from “Station” at “Master”. Sends a good response even if “Station” did not have a call request on “Master”. Specifying “Master” or “Station” as 0 is a syntax error. If the panic alarm call request is not present (i.e., not active or acknowledged) then this will return a good response (as if it was actually cancelled).

Examples:

Pcan 10 1130 *Cancel panic alarm call request from station 1130 to master 10.*

Status Messages:

Reports that a panic alarm call request from “Station” was canceled at “Master”.

Examples:

Pcan 10 1130 *Panic alarm call request from station 1130 cancelled at master 10.*

Pcrq – Panic Alarm Call Request

Register Function Code: 247

Parameters:

Master
Station

Implementation Status:

Command/response and status messages implemented in DXL version 2.3.0.

Command/Response Messages:

Causes a panic alarm station call request alarm from "Station" to be queued at "Master". The "Master" that this panic alarm call request is queued is specified as "Master". This always responds with a good response even though it may (if the station's panic alarm call request is disabled) not queue the panic alarm call request at the "Master" but instead may queue it at the secondary master of "Master". If this command is successful, then the status message is generated to indicate that the command has been processed (if the call request is not disabled, see **Enbs**).

Examples:

Pcrq 1 1130 *Panic alarm call request from station 1130 will be queued at master 1.*
Done Pcrq 1 *1130 Response to panic alarm call request from station 1130 queued.*

Status Messages:

Reports that a panic alarm call request from "Station" was queued at "Master". This can also report the station panic alarm call requests queued on "Master" when the **Stat** or **AllS** command is sent. All station panic alarm call requests on "Master" will be reported with this status message.

Examples:

Pcrq 10 1130 *Panic alarm call request from station 1130 queued at master 10.*

Pend – End Page To Zone

Register Function Code: 202

Parameters:

Master
Zone

Implementation Status:

Command/response messages implemented in DXL version 0.1.0b1.
Status message implemented in DXL version 0.5.3b3.

Command/Response Messages:

This command is used to end the call on the "Master". Specifying a "Master" of 0 is a syntax error. If "Zone" is 0, then any existing zone being paged will be ended. If "Zone" is not 0, then "Master" will only be disconnected if it is connected to "Zone" or if it was not connected to anything. The response will be generated even if this master was not previously connected to anything (i.e., it returns a good response whether it actually does any disconnects or not). This will only end calls, not monitors.

Examples:

Pend 10 0 *Ends current page, if any, on master 10*
Done Pend 10 99 *Response to above command if paging zone 99*
Pend 10 99 *Ends current page on master 10 but only if from zone 99*
Done Pend 10 99 *Response to above command if paging zone 99*

Status Messages:

When a new connection is made without ending the previous one (i.e., suppose the user does two consecutive Pages to different zones) a status message will be generated. This will be generated to tell the host which zone call has been ended. This will not be generated if the call ended is a page. If there is an **EndC** used to end the page this will not be generated.

Examples:

Pend 10 1130 *Current call from master 10 to zone 1130 was ended*

Priv – Set Station's Privacy Mode

Register Function Code: 252

Parameters:

Station
Privacy Mode (0/1)

Implementation Status:

Status messages implemented in DXL version 3.1.0b1.

Command/Response Messages:

N/A.

Status Messages:

This message reports whenever a station is in put into or taken out of privacy mode. When the *Privacy Mode* is 0 then the station is not in privacy mode and when it is 1 then the station is in privacy mode. This can also be generated when the **Stat 0** or **AllS** command is sent. Only stations that are in privacy mode will be reported in response to the **Stat** or **AllS** command.

Examples:

Priv 1130 1 *Station 1130 has been put into privacy mode*
Priv 1130 0 *Station 1130 has been taken out of privacy mode*

PVad – Page Zone Volume Adjustment

Register Function Code: 92

Parameters:

Zone
Increment

Implementation Status:

Command/response messages implemented in DXL version 0.1.0b1.

Command/Response Messages:

This command changes the volume of "Zone". If "Increment" is 0 (zero), then the zone's volume will decrement by 1 setting (i.e., get quieter). If "Increment" is 1 (one), then the zone's volume will increment by 1 setting (i.e., get louder. Note that the good response is always sent back, regardless of whether or not the volume can increase or decrease any more than its current setting (i.e., whether it is at its maximum or minimum respectively).

Examples:

PVad 1130 1 *Increment zone 1130's volume by 1*
PVad 1130 0 *Decrement zone 1130's volume by 1*

Status Messages:

N/A.

RmMG – Remove Station Group from Intercom Auto-Monitor List

Register Function Code: 240

Parameters:

Master
Station Group

Implementation Status:

Command/response and status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

This command is sent to stop monitoring a “Station Group” from “Master”. The ”Station Group” is removed from the master’s monitor list (which is cycled through on the master) instead of doing a dedicated monitor as the Imon command does. Passing 0 as the “Station Group” will remove all stations from the master’s monitor list.

Examples:

RmMG 1 99 *Removed station group 99’s members from master 1’s monitor list.*

Status Messages:

Sent when the monitor list of a master is changed from within the DXL system.

Examples:

RmMG 1 99 *Master 1 removed station group 99’s members from its monitor list.*

RmMS – Remove Station from Intercom Auto-Monitor List

Register Function Code: 91

Parameters:

Master
Station

Implementation Status:

Command/response and status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

This command is sent to stop monitoring a "Station" from "Master". The "Station" is removed from the master's monitor list (which is cycled through on the master) instead of doing a dedicated monitor as the Imon command does. Passing 0 as the station will remove all stations from the master's monitor list.

DXL version 3.0.1b3: If there is no station with this number then the "Station" number will be looked up to see if there is a station group which has this number. If there is then this will remove the station group's members from the master's monitor list, as if the user entered the **RmMG** command with the "Master" and "Station" as the parameters.

Examples:

RmMS 1 99 *Removed station 99 from master 1's monitor list.*

Status Messages:

Sent when the monitor list of a master is changed from within the DXL system.

Examples:

RmMS 1 99 *Master 1 removed station 99 from its monitor list.*

Sadd – Add Station To Zone

Register Function Code: 253

Parameters:

Zone
Station

Implementation Status:

Command/response and status messages implemented in DXL version 5.0.0b1.

Command/Response Messages:

Attempts add a station to a zone. The “Station” is added to the “Zone” (i.e., any page to the “Zone” will connect the “Station”). This can be done while a “Zone” is in a call, signal, or any other type of connection. If the “Station” is already part of the first “Zone” then this will return a good response. Passing the “Station” as zero will remove all stations which have been previously added to the “Zone”, essentially restoring it to its original configuration.

“Station” can not be a compound station, it must be a single station.

Examples:

Sadd 1 2 *Add station 2 to zone 1*

Status Messages:

Reports that the “Station” has been added into the “Zone”. This can also be generated by the **Stat** or **AllS** command.

Examples:

Sadd 1 2 *Station 2 has been added into zone 1*

SetG – Set Master Station for Station Group

Register Function Code: 246

Parameters:

Station Group
Master

Implementation Status:

Command/response messages implemented in DXL version 2.3.0. There is, unlike **SetM**, no status messages associated with this command[35].

Command/Response Messages:

Change the master that a station group's members reports their call requests, panic alarms and/or tamper alarm switches to. These will be re-routed to the specified master. Note that this not only affects the switches themselves but, if there are any existing call requests, panic alarms and/or tamper alarms, they will be re-routed as a result of this command[36]. Passing 0 as the "Master" will cause the "Station Group" to revert to its original (i.e., configuration) routing for all alarms.

Examples:

SetG 1 10 *Change station group 1's call requests to report to master 10*

Status Messages:

N/A.

[35] The reason there is no status message sent when the Stat command is sent is because of the nature of groups. If two groups have common stations, then reporting a SetG would not allow the user to determine which master the common stations were reporting to. To prevent this from happening, the Stat command will result in SetMs for all the stations in the group(s).
[36] Unlike the DXI which did not re-route existing alarms when the SetM command was issued. The DXL will re-route the existing alarms to the appropriate masters .

SetM – Set Master Station for Intercom Station

Register Function Code: 44

Parameters:

Station
Master

Implementation Status:

Command/response and status messages implemented in DXL version 2.3.0.

Command/Response Messages:

Change the master that a station reports its call request and/or tamper alarm switches and/or panic alarm switches to. These will be re-routed to the specified master. Note that this not only affects the switches themselves but, if there are any existing call requests and/or tamper alarms and/or panic alarms, they will be re-routed as a result of this command[37]. Passing 0 as the "Master" will cause the "Station" to revert to its original (i.e., configuration) routing for all alarms.

DXL version 3.0.1b3: If there is no station with this number then the "Station" number will be looked up to see if there is a station group which has this number. If there is a station group numbered "Station" then this command will change the station group's members audio level alarm levels as if the host sent a **SetG** command with "Station" and "Master".

Examples:

SetM 1 10 *Change station 1 to report to master 10*

Status Messages:

This is sent in response to a **Stat 0** or **AllS** command. If the station has never had its routing changed via a **SetM** or **SetG** then this will not be reported. If it has been changed then the **SetM** will report the new routing information associated with this station.

Examples:

SetM 1 10 *Reports that someone changed station 1 to report to master 10*

[37] Unlike the DXI which did not re-route existing alarms when the SetM command was issued. The DXL will re-route the existing alarms to the appropriate masters .

SetO – Set Station Output

Register Function Code: 33

Parameters[38]:

Station
State (1-7)

Implementation Status:

Command/response messages partially implemented in DXL version 2.1.0. These can be sent but there is no way to define a configuration that has a station output whose function is not system controlled. Hence they can be sent to the DXL and the Done reply received without anything happening.
The syntax of this message is subject to change in future releases.

Command/Response Messages:

This message tells "Station" to set its LED to a particular "State. "State" is encoded as follows:

STATE	OUTPUT	REMARKS
0	OFF	FULLY OFF.
1	ON	FULLY ON.
2	WINK	7/8 DUTY CYCLE.
3	SLOW FLASH	½ DUTY CYCLE
4	FLASH	½ DUTY CYCLE WITH QUICK FLASH
5	BLINK	1/8 DUTY CYCLE

The "pulse time" referred to in state 7 is configurable for each station. This is part of the station configuration and can not be changed from the host. This overrides the station's preprogrammed output for the station.

Examples:

SetO 1130 1 *Turn on station 1130's LED*
SetO 1130 6 *Turn off station 1130's LED*

Status Messages:

N/A.

[38] The parameters for this command differ from the corresponding DXI command.

SetS – Set Secondary Master (of Primary Master)

Register Function Code: 43

Parameters:

Master
Secondary Master

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.3b5.

Command/Response Messages:

Change the secondary master of the first "Master" from its current setting to "Secondary Master". There is some error checking done to ensure that the masters secondary does not violate some secondary rules (i.e., a master can not be a secondary of itself). This basically changes DXL call request routing tables while the system is running. "Secondary Master" can be a group of masters or a single master. Reception of this command may cause alarm to be transferred (i.e., existing secondary alarms may be transferred to a new secondary master). These new alarms will occur after the command's response has been set. It is valid to change "Master" to have a "Secondary Master" of 0 (i.e., no secondary master). Changing the secondary master to the same as the existing secondary master will result in a good response.

Examples:

SetS 1 10 *Change master 1's secondary to master 10*

Status Messages:

Reports that the secondary master of "Master" has been changed to "Secondary Master". This can also be generated in response to a **Stat** or **AllS** command. This will only be sent if the secondary master differs from the configuration's secondary master for this master.

Examples:

SetS 1 10 *Reports that master 1's secondary was changed to master 10*

Sgnl – Start Signal (Tone to a Zone)

Register Function Code: 24

Parameters:

Master
Signal

Implementation Status:

Command/response and status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

This command starts the "Signal" and lets it run until it is ended with an **EndS** or the tone itself completes (i.e., it is defined to be a finite tone). Finite tones can be ended with an **EndS** or they can terminate normally. The "Master" is optional (can be 0). If "Master" is not 0, then only this master can stop the tone. If "Master" is 0, then any master in the system can stop the signal. If "Signal" is already running this will not restart "Signal" (i.e., disconnect and reconnect it)."Signal" can not be 0. A good response is returned if the tone is successfully started on at least 1 station in the zone.

Examples:

Sgnl 10 99 *Start signal 99*

Status Messages:

This message indicates that "Signal" was started from the DXL. "Master" will never be 0 when this command is generated by the host. Only signals which are in progress at the master will be reported by the **Stat** or **AllS** command.

Examples:

Sgnl 10 99 *Master 10 started signal 99*

SMas – Copy Master Music Selection To Station

Register Function Code: 65

Parameters:

Master
Station

Implementation Status:

Command/response messages implemented in DXL version 2.1.0.

Command/Response Messages:

This command is sent to change the music on "Station" to the same channel as what is currently on "Master". If there is no music on "Master" then there will be no music placed on "Station" (if it was connected to a music channel, it will be disconnected). If there is music on "Master" then it will be placed on "Station" (this can result in the music on "Station" being changed). If the music on "Station" was the same as on "Master" then this will return a good response. If there is no music on "Master" and no music on "Station" then this will return a good response as well.

DXL version 3.0.1b3: If there is no station with this number then the "Station" number will be looked up to see if there is a station group which has this number. If there is a station group numbered "Station" then this command will change the station group's members music selections as if the host sent a **GMas**command with "Master" and "Station".

Examples:

SMas 10 1130 *Change station 1130's music channel to the same as master 10*

Status Messages:

N/A.

SMus – Station Music Selection

Register Function Code: 64

Parameters[39]:

Station
Music
Low/High Volume

Implementation Status:

Command/response messages implemented in DXL version 0.5.3b3.
Status message implemented in DXL version 3.0.1b3.

Command/Response Messages:

Sent from the host to tell "Station" to change its music channel to "Music". This can be done even when "Station" is in a call (it is assumed that music is lower priority than calls so the actual music will not be present until the call is ended). When "Music" is 0, then the music is disconnected from "Station" (the "Low/High Volume" should be zero in this case). This returns a good response if "Music" is the same channel that is currently connected. The "Low/High Volume" is used to select the appropriate volume level of the music channel, zero meaning select the low volume and non-zero meaning select the high volume.

DXL version 3.0.1b3: If there is no station with this number then the "Station" number will be looked up to see if there is a station group which has this number. If there is a station group numbered "Station" then this command will change the station group's members music setting as if the host sent a **GMus**command with ""Station", "Music" and Low/High Volume".

Examples:

SMus 1130 2 1 *Change station 1130 to music channel 2, high volume*
Smus 1130 20 0 *Change station 1130 to music channel 20, low volume*

Status Messages:

This will be sent in response to a **Stat** or **AllS** command. Only stations which have music will be reported in response to the **Stat** or **AllS** command.

Examples:

SMus 1130 1 0 *Music channel of station 1101 was changed to music channel 1, low*
SMus 1130 0 0 *Music channel of station 1130 was disconnected*

[39] The parameters for this command differ from the corresponding DXI command.

Sntx – Failure Due to Syntax Error

Register Function Code: 204

Parameters:

Function Code and Parameters.

Implementation Status:

Response message implemented in DXL version 0.1.0b1.

Response Message:

This response message indicates that the command failed to be parsed correctly. The parameter string is an echo of the command which caused the problem. This can result from invalid ID numbers or a badly formatted command (i.e., missing parameters). A badly formatted command can include commands who have permission problems. For example, an "Ical 1 1130" would return a syntax error if master 1 and/or station 1130 did not exist or if master 1 did not have station 1130 on its permission list. If the Ical command was disabled in the configuration then this would also generate a syntax error.

Examples:

Sntx Ical 1130 *Command "Ical 1130" failed.*

Status Messages:

N/A.

Ssub – Remove a Station From Zone

Register Function Code: 254

Parameters:

Zone
Station

Implementation Status:

Command/response and status messages implemented in DXL version 5.0.0b1.

Command/Response Messages:

Attempts remove "Station" from the "Zone (but only if the zone had previously had this station added via the **Sadd** command). This can be done while a "Zone" is in a call, signal, or any other type of connection. If the "Station" is already part of the "Zone" then this will return a good response. Passing the "Station" as zero will remove all stations which were added via the **Sadd** command, essentially restoring it to its original configuration.

"Station" can not be a compound station, it must be a single station.

Examples:

Ssub 1 2 *Remove the station 2 from zone 1*

Status Messages:

Reports that the "Station" has been removed from the "Zone".

Examples:

Zsub 1 2 *Station 2 has been removed from zone 1*

Stat – Request Master Status

Register Function Code: 12

Parameters:

Master

Implementation Status:

Command/response messages implemented in DXL version 0.1.0b1. The **Enbl** is status was added to the response to the this request in DXL version 0.5.3b1.

Command/Response Messages:

This tells "Master" to report its current status to the host. This will cause **Mcal**, **Ical**, **Page**, etc. status messages if "Master" is involved in a call or **Imon** status messages if "Master" is involved in a monitor. All call requests and faults queued at "Master" will be sent to the host using the appropriate **Icrq**, **Halm**, etc. status message. It will also report the **Enbl** status messages of "Master" (see **Enbl** status message). The order the individual status messages are sent is not specified. Only messages which are not defaults (i.e., when the master starts up it has no music therefore, if there is no music on the station, the music status message will not be sent) are sent. The host may clear its status when it sends this message so all appropriate messages to synchronize the host must be sent. Sending "Master" of 0 will update all global (i.e., not associated with a specific master) messages, like visiting booth connections, etc.

Examples:

Stat 10 *Report the status of master 10*

Status Messages:

N/A.

Tack – Tamper Alarm Acknowledge

Register Function Code: 213

Parameters:

Master
Station

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.5.

Command/Response Messages:

Causes a tamper alarm from "Station" to be acknowledged by "Master". If the tamper alarm is already acknowledged then this will return a good response. If the tamper alarm is not active or already acknowledged, then this will return a fail response.

Examples:

Tack 10 1130 *Acknowledge station 1130's tamper alarm on master 10*

Status Messages:

Reports that a tamper alarm from "Station" was acknowledged by "Master". This can also be sent in response to a **Stat** or **AllS**. Only stations which have acknowledged tamper alarms will be reported in response to the **Stat** or **AllS** command.

Examples:

Tack 10 1130 *Station 1130's tamper alarm acknowledged by master 10*

Talm – Tamper Alarm

Register Function Code: 212

Parameters

Master
Station

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.5.

Command/Response Messages:

Causes a station tamper alarm from "Station" to be queued at a master. The master that the alarm is queued at is the station's master (i.e., where it queues its tamper alarms), not the "Master". Therefore, it is legal to pass 0 as the "Master". This never sends a good response and may (if the station's tamper alarm is disabled) not queue the tamper alarm at the station's master. If this command is successful, then the status message is generated to indicate that the command has been processed.

Examples:

Talm 10 1130 *Queue a tamper alarm from station 1130 on master 10*

Status Messages:

Reports that a tamper alarm on "Station" and the alarm has been queued on "Master". This can also be sent in response to a **Stat** or **AllS** command. Only stations with active, unacknowledged tamper alarms will be reported in response to the **Stat** or **AllS** command. These can be suppressed with the **EnbT**host command.

Examples:

Talm 10 1130 *Tamper alarm on station 1130 queued on master 10*

Tcan – Tamper Alarm Cancel

Register Function Code: 214

Parameters:

Master
Station

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.5.

Command/Response Messages:

Cancels the tamper alarm from “Station” at “Master”. Sends a good response even if “Station” did not have a tamper alarm on “Master”. Specifying “Master” or “Station” as 0 is a syntax error. If the alarm is not present (i.e., not active or acknowledged) then this will return a good response (as if it was actually cancelled).

Examples:

Tcan 10 1130 *Cancel tamper alarm from station 1130 to master 10*

Status Messages:

Reports that a tamper alarm from “Station” was canceled at “Master”.

Examples:

Tcan 10 1130 *Tamper alarm from station 1130 cancelled at master 10*

Time – Set System Time

Register Function Code: 6

Parameters:

Hour
Minute
Second

Implementation Status:

Command/response and status messages implemented in DXL version 0.5.3b3.

Command/Response Messages:

This command is used to set the time on the DXL's real time clock. The time is always in 24 hour format, in the format HH MM SS, ranging from 00 00 01 to 23 59 59. Any errors in the format of the time is considered a syntax error. Otherwise, if the clock is changed, then a good response will be sent[40]. Note that we don't allow the time to be set to 00:00:00 (midnight) because of potential race conditions with the **Date** command. There is currently a one second "dead" band around the clock setting (so we don't allow it to be set to 00:00:00). We may want to increase this "dead" band at some point (the amount of time depends upon the potential delay between a **Date** and **Time** command.

Examples:

Time 22 15 39 *Set time to 22:15:39*

Status Messages:

This message will reports that the DXL's clock has changed. When the clock is set on another host and it propagates to the local host, then this message will be generated by the local host (i.e., if there are multiple hosts defined then receiving the **Time** command on one host will cause **Time** status messages to be sent to the other hosts). However, if the clock was changed due to a **Time** command on this host, then this status message will not be generated, just the response.

Examples:

Time 22 15 39 *Time was changed to 22:15:39*

[40] The Date and Time commands are independent of each other and, as such, there is a possible race condition around midnight. The Date command should always be set after the time to prevent this race. If the date is set and then midnight occurs before the time is set, then the date will change and the clock will be incorrect. By sending the time after the date, this can be prevented.

Vcal – Visiting Booth Station Call[41]

Register Function Code: 69

Parameters:

Master
Visiting Station

Implementation Status:

Command/response and status messages implemented in DXL version 2.0.1b3.

Command/Response Messages:

Attempts a call from "Master" to "Visiting Station" which should be in a user defined visiting booth (see **Vset**). Specifying a "Visiting Station" of 0 will disconnect the current call (see **Iend/Mend/Pend**). If "Master" is in a call to something else (i.e., a master, station, etc.) then this will disconnect that call (and generate an **Iend/Mend/Pend** for "Master") before trying to connect to "Visiting Station". The **Iend/Mend/Pend**, in this case, will always be sent before the response. If "Master" was doing a monitor, then that will be disconnected (and generate an **EndM** for "Master") before trying to connect to "Visiting Station". The **EndM**, in this case, will always be sent before the response. Specifying "Master" as 0 is a syntax error. The good response will be generated if the connection is made or the device to which "Master" was previously calling was "Visiting Station". In the second case, the disconnection is not done and "Master" and "Visiting Station" are left connected. If the "Visiting Booth" is busy in a higher priority call, then this command will generate a busy response.

Examples:

Vcal 10 1130 *Call visiting station 1130 from master 10.*

Status Messages:

Reports that "Visiting Station" has been connected to "Master". This can also report what station (if any) "Master" is connected to when the **Stat** or **AllS** command is sent. If no station is connected to this master, then the message is not generated (i.e., is not sent with a station of 0).

Examples:

Vcal 10 1130 *Visiting Station 1130 was called from master 10.*

[41] This is similar to **Bcal**except that this command is for dynamic (i.e., host defined) visiting booths, not pre-defined visiting booths (i.e., defined in configuration).

Vend – End Call To Visiting Booth Station[42]

Register Function Code: 235

Parameters:

Master
Visiting Station

Implementation Status:

Command/response and status messages implemented in DXL version 2.0.1b3.

Command/Response Messages:

This command is used to end the call on the "Master". Specifying a "Master" of 0 is a syntax error. If "Visiting Station" is 0, then any existing visiting booth call will be ended (if there was a page or master call, then this would not be disconnected and this would return a failure). If "Visiting Station" is not 0, then "Master" will only be disconnected if it is connected to "Visiting Station". The response will be generated even if this master was not previously connected to anything (i.e., it returns a good response whether it actually does any disconnects or not). This will only end calls, not monitors.

Examples:

Vend 10 0 *Ends current call, if any, on master 10*
Done Vend 10 30 *Response to above command if connected to visiting station 30*
Vend 10 30 *Ends current call on master 10 but only if from visiting station 30*
Done Vend 10 30 *Response to above command if connected to visiting station 30*

Status Messages:

When a new connection is made without ending the previous one (i.e., suppose the user does two consecutive **Vcal**s to different stations) a status message will be generated. This will be generated to tell the host which station call has been ended. This will not be generated if the call ended is not to a station. If there is an **EndC** used to end the call this will not be generated.

Examples:

Vend 10 1130 *Current call from master 10 to visiting station 30 was ended*

[42] This is similar to **Bend** except that this command is for dynamic (i.e., host defined) visiting booths, not pre-defined visiting booths (i.e., defined in configuration).

Vmon – Monitor Visiting Booth From Master[43]

Register Function Code: 70

Parameters:

Master
Visiting Station

Implementation Status:

Command/response and status messages implemented in version 2.0.0.

Command/Response Messages:

Sent to tell "Master" to monitor "Visiting Station". If "Visiting Station" is 0, then "Master" will stop monitoring whatever it was previously (if anything) monitoring, regardless of whether it was a station or not. This is the same as the **EndM** command. If "Master" was not monitoring anything, then this will result in a good response. If "Visiting Station" is not 0, then this "Master" will monitor "Visiting Station". If "Master" was already monitoring "Visiting Station" then this will result in a good response but the connection will be left. If "Visiting Station" is not 0 and this master was monitoring some other device, this device will be disconnected (which will result in an **Imon/Vmon/Bmon X 0** status message being sent before the **Vmon** response is generated).

Examples:

Vmon 10 1130 *Monitor visiting station 1130 on master 10*
Vmon 10 0 *Stop monitoring on master 10*

Status Messages:

Reports that "Master" is now monitoring "Visiting Station". This can also be sent in response to a **Stat** or **AllS** command (if a monitor was in progress). This is only reported (in response to the **Stat** or **AllS** command) if the visiting booth being monitored is not 0.

Examples:

Vmon 10 1130 *Master 10 is now monitoring visiting station 10*

[43] This is similar to **Bmon** except that this command is for dynamic (i.e., host defined) visiting booths, not pre-defined visiting booths (i.e., defined in configuration).

VRec – Record Visiting Booth[44]

Register Function Code: 94

Parameters[45]:

Call Recorder
Visiting Station

Implementation Status:

Command/response messages implemented in DXL version 2.0.0.

Command/Response Messages:

Connects "Visiting Station" to the "Call Recorder". "Visiting Station" is the visiting station specified when this visiting booth was created. This will return a bad return code if the visiting station does has not been created with a **Vset**. "Call Recorder" can not be 0. If the "Visiting Station" is 0, then the call recorder is disconnected (from whatever it was connected to, even if that was not a visiting station, exactly like doing an **EndR**). If "Visiting Station" was already connected to the "Call Recorder", then this will return a good response (if "Call Recorder" is not 0). If "Visiting Station" is not connected (and "Call Recorder" is 0) then this will also return a good response. If "Call Recorder" was connected to any other device, then this will be disconnected and an **EndR** command will be generated before this command's response is received by the host.

Examples:

VRec 1 10 *Record visiting station 10 on call recorder 1*

Status Messages:

Reports that "Visiting Station" is being recorded by "Call Recorder". This can be also be sent in response to a **Stat** or **AllS** command. Only call recorders which are recording stations will be sent this messages in response to the **Stat** or **AllS** command.

Examples:

VRec 1 10 *Visiting station 10 is being recorded by call recorder 1*

[44] This is similar to **BRec** except that this command is for dynamic (i.e., host defined) visiting booths, not pre-defined visiting booths (i.e., defined in configuration).
[45] The parameters for this command differ from the corresponding DXI command.

Vset – Start Visiting Booth (With/Without Timeout) [46]

Register Function Code: 66

Parameters:

Master
Visiting Station
Visiting Station
Timeout (minutes)

Implementation Status:

Command/response and status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

Activate a "Visiting Station" to "Visiting Station" connection. If "Timeout" is 0 then the visiting booth connection will not timeout (i.e., it can last forever if necessary). If "Timeout" is not 0, then this will limit the connection to a maximum of that length of time (the connection could be prematurely ended by one of the visiting stations hanging up, if they have handsets). If the two "Visiting Stations" were already active, then this command will return a good response (as if they had just been made active) but will reprogram the timer to the value passed in the new command. If either or both "Visiting Stations" are busy, a busy response will be generated. If the "Visiting Stations" have handsets then the connection will not be made until they go off hook (if they were both idle and off hook at the time the command was sent, then the connection is made immediately). The 'Master' can be specified as 0 or any master assigned this host.

Examples:

Vset 1 5 10 0 *Start never-ending visiting booth with stations 5 and 10 from master 1*
Vset 1 5 10 20 *Start 20 minute visiting booth with stations 5 and 10 from master 1*

Status Messages:

This reports that a visiting booth call has started. This can also be generated by a **Stat** or **AllS** command. Only visiting booths that are active will be reported in response to the **Stat** or **AllS** command. The amount of time left (rounded up to the nearest minute) will be sent in this case.

Examples:

Vset 1 5 10 0 *A never-ending visiting booth connection was started from master 1*
Vset 1 5 10 20 A *20 minute call visiting booth connection was started from master 1*

[46] This is similar to **Bset** except that this command is for dynamic (i.e., host defined) visiting booths, not pre-defined visiting booths (i.e., defined in configuration).

VstB - Visiting Booth State

Register Function Code: 68

Parameters:

Master
Visiting Station
State number

Implementation Status:

Status message implemented in DXL version 2.1.1b3.

Status Messages:

This status message is sent to the host computer when a visiting booth changes it's state. The state of the visiting booth roughly corresponds to the message printed out on it's video display. The states are numbered 1-9 (inclusive) They represent the following:

State number	Description	Video display/connection
1[47]	Idle	No text
3	Assigned, on hook	Lift Handset To Start
4	Assigned, off hook	Please Wait
5	Ending Call	End of Visit
8	In Call	Connection to visitor/inmate

The Idle state is the unconnected, unassigned visiting booth state. When a master assigns a pair of visiting stations, then the visiting stations will enter either of the assigned states, depending upon whether it is on hook or off hook. When both visiting stations go off hook, then the visiting stations will enter the In Call state, at which time the video will be connected. When the call times out or is ended by the master, then the visiting booths go to the ending state for 1 minute, after which the visiting booths return to the idle state. When the master calls a visiting booth, the visiting booth displays either Master Calling states, depending upon whether the visiting booth is on or off hook.
.

Examples:

VstB1 5 1 *Reports the current state of the visiting booth as 1 (idle, on hook).*

[47] In the DXI, there was a differentiation between stations that were idle on hook and stations that were idle off hook. The DXL makes no such distinction, hence there is only state 1, not state 2. All these states are compatible with the DXI states, hence the strange numbering of the states (i.e., missing 6, 7 and 9).

Vstp – Stop Visiting Booth Connection[48]

Register Function Code: 67

Parameters:

Master
Visiting Station
Visiting Station

Implementation Status:

Command/response and status messages implemented in DXL version 1.1.0b4.

Command/Response Messages:

Sent from the host to deactivate "Visiting Station" connection. If "Visiting Station" connection was not active (i.e., the two "Visiting Stations" were not involved in a visiting booth connection), then this will return a good return code (i.e., it acts like it was just disconnected). If the "Visiting Stations" were in a visiting booth connections with other stations (but not each other) then a fail response will be generated. The 'Master' can be specified as 0 or any master assigned this host. Note that the visiting stations must be passed in the same order that the **Vset** command was sent (i.e., can't send **Vset** 1 2 3 and **Vstp** 1 3 2).

Examples:

Vstp 1 5 10 *Stop visiting booth between stations 5 and 10 from master 1*

Status Messages:

Reports when "Visiting Booth" is disconnected. This typically happens when the timeout occurs.

Examples:

Vstp 1 5 10 Visiting booth has ended (i.e., timed out or hung up) from master 1

[48] This is similar to **Bstp** except that this command is for dynamic (i.e., host defined) visiting booths, not pre-defined visiting booths (i.e., defined in configuration).

Zadd – Add Zone Members To Zone

Register Function Code: 242

Parameters:

Zone
Zone

Implementation Status:

Command/response and status messages implemented in DXL version 2.0.1b3.

Command/Response Messages:

Attempts merge two zones into one. The members of the second “Zone” are merged into the first “Zone” (i.e., any page to the first “Zone” will connect the second “Zone”). This can be done while a “Zone” is in a call, signal, or any other type of connection. If the second “Zone” is already part of the first “Zone” then this will return a good response. If the first and second “Zone” are the same then this is an error and will return a bad return code. Passing the second “Zone” as zero will disconnect all zones which are currently merged with the first “Zone”, essentially restoring it to its original configuration.

Examples:

Zadd 1 2 *Add the members of zone 2 to zone 1*

Status Messages:

Reports that the second “Zone” has been merged into the first “Zone”. This can also be generated by the **Stat** or **AllS** command.

Examples:

Zadd 1 2 *Zone 2’s members have been merged into zone 1*

ZMas – Copy Master Music Selection To Zone

Register Function Code: 63

Parameters:

Master
Page Zone

Implementation Status:

Command/response messages implemented in DXL version 2.1.0.

Command/Response Messages:

This command is sent to change the music on “Zone” to the same channel as what is currently on “Master”. If there is no music on “Master” then there will be no music placed on “Zone” (if it was connected to a music channel, it will be disconnected). If there is music on “Master” then it will be placed on “Zone” (this can result in the music on “Zone” being changed). If the music on “Zone” was the same as on “Master” then this will return a good response. If there is no music on “Master” and no music on “Zone” then this will return a good response as well.

Examples:

ZMas 10 99 *Change zone 99’s music channel to the same as master 10*

Status Messages:

N/A.

ZMus – Page Zone Music Selection

Register Function Code: 62

Parameters[49]:

Zone
Music
Low/High Volume

Implementation Status:

Command/response messages implemented in DXL version 0.5.3b3.
Status message implemented in DXL version 3.0.1b3.

Command/Response Messages:

Sent from the host to tell "Zone" to change its music channel to "Music". This can be done even when "Zone" is in a page (it is assumed that music is lower priority than pages so the actual music will not be present until the page is ended). When "Music" is 0, then the music is disconnected from "Zone" (the "Low/High Volume" should be zero in this case). This returns a good response if "Music" is the same channel that is currently connected. The "Low/High Volume" is used to select the appropriate volume level of the music channel, zero meaning select the low volume and non-zero meaning select the high volume.

Examples:

ZMus 99 2 1 *Change zone 99 to music channel 2, high volume*
ZMus 99 20 0 *Change zone 99 to music channel 20, low volume*

Status Messages:

This will be sent in response to a **Stat** or **AllS** command. Only zones which have music will be reported in response to the **Stat** or **AllS** command.

Examples:

ZMus 1130 1 0 *Music channel of zone 1101 was changed to music channel 1, low*
ZMus 1130 0 0 *Music channel of zone 1130 was disconnected*

[49] The parameters for this command differ from the corresponding DXI command.

Zset – Start Station to Zone Page

Register Function Code: 244

Parameters:

Station
Zone

Implementation Status:

Command/response and status messages implemented in DXL version 2.0.1b3.

Command/Response Messages:

Attempts a page from the "Station" to the "Zone". If "Station" is in a call to something else (i.e., a master, station, another zone, etc.) then this will disconnect that call (and generate an **Zstp** for "Station") before trying to connect to zone. The **Zstp**, in this case, will always be sent before the response. The good response will be generated if the connection is made or if connection had previously been made. In the second case, the disconnection is not done and the "Station" and "Zone" are left connected. If either the "Station" or "Zone" is busy in a higher priority call, then this command will generate a busy response.

Examples:

Zset 10 1130 *Page zone1130 from station 10*

Status Messages:

Reports that the "Station" has been connected to the "Zone". This can also be generated by the **Stat** or **AllS** command. Only stations involved in a call are reported in response to the **Stat** or **AllS** command.

Examples:

Zset 1101 1130 S*tation 1101 paged zone 1130*

Zstp – End Station To Zone Page

Register Function Code: 245

Parameters:

Station
Zone

Implementation Status:

Command/response and status messages implemented in DXL version 2.0.1b3.

Command/Response Messages:

This command is used to end the page between a “Station” and its “Zone”. If “Station” is not connected to anything, then this will return a good response. If the “Station” was paging “Zone” then they will be disconnected and a good response returned. If the “Station” is involved in another call, this will generate a fail response. Specifying “Station” as 0 is an error.

Examples:

Zstp 1101 1130 *End station 1101 to zone 1130 page*

Status Messages:

Reports when page from “Station” to “Zone” is ended.

Examples:

Zstp 1101 1130 *Connection from station 1101 to zone 1130 has ended*

Zsub – Remove Zone Members From Zone

Register Function Code: 243

Parameters:

Zone
Zone

Implementation Status:

Command/response and status messages implemented in DXL version 2.0.1b3.

Command/Response Messages:

Attempts separate two zones from one. All the members of the second "Zone" are removed from the first "Zone" (but only if the zone had previously had its members added via the **Zadd** command). This can be done while a "Zone" is in a call, signal, or any other type of connection. If the second "Zone" is already part of the first "Zone" then this will return a good response. If the first and second "Zone" are the same then this is an error and will return a bad return code. Passing the second "Zone" as zero will disconnect all zones which are currently merged with the first "Zone", essentially restoring it to its original configuration.

Examples:

Zsub 1 2 *Remove the members of zone 2 from zone 1*

Status Messages:

Reports that the second "Zone" has been removed from the first "Zone".

Examples:

Zsub 1 2 *Zone 2's members have been removed from zone 1*