Global Connect TCP API Specification

Version 1.0

TABLE OF CONTENTS

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

1. GLOBAL CONNECT – MODULAR I/O DESIGN CONCEPT

The Global Connect product family features a modular design that offers infrared, serial, relay, and HDMI switch/control capabilities. Global Connect's chassis design provides slots for up to 10 modules, where each module occupies 1 or more slots and is handled as a separate device and TCP endpoint having its own unique IP address.

In terms of using the API, the individual addressability of Global Connect offers two significant benefits. First, it allows for a consistent API regardless of how many control ports of each type are contained within a chassis. Second, it allows for backwards compatibility with the TCP API used in earlier product lines.

2. NETWORK CONFIGURATION

Each Global Connect device has a network interface as the first (leftmost) module, available as either 1-port Ethernet RJ45 100Mbps (1 slot), or 2-port PoE Ethernet RJ45 100 Mbps switch (2 slots). The Ethernet network module requires no configuration, and therefore is not addressed via the API.

Global Connect modules are configured using the embedded configuration web page, or via API request. Factory default settings can be applied by selecting the "reset" option on the configuration web page, or by pressing a module's first (leftmost) LED/button for 10 or more seconds, or by API request. This allows any module to be reset to a known configuration and IP address, allowing access for configuring with desired settings.

By default, a Global Connect module is set to automatically acquire its IP configuration via DHCP. However, if a DHCP server is not available, the module falls back to the default static IP address of 192.168.1.<70+n>, where 'n' is the module's address (which, by default, starts at 1 for the leftmost module, and increases sequentially for each module to the right).

3. NETWORK DISCOVERY

Global Connect provides two methods of network discovery – periodic UDP multicast message and multicast DNS (mDNS).

Discovery and management of Global Caché devices can be achieved using the *iHelp* utility available for download at http://www.globalcache.com/downloads. iHelp has several useful functions, including the ability to discover and display all Global Caché devices connected to the network. This is accomplished through a mechanism wherein devices send periodic network messages called "beacons, that contain identifying information such as IP address, firmware version, and MAC ID. iHelp receives the beacons and displays the information to the user in a list of discovered devices. iHelp also provides additional functions, such as updating devices. Please refer to iHelp documentation for further details.

The periodic beacons sent by Global Connect modules are actually UDP messages, sent to the multicast IP address 239.255.250.250, multicast group destination MAC 01:00:5E:7F:FA:FA, and destination port 9131. Beacons are sent shortly after a successful network connection and continue at regular intervals of 10 seconds. Any network device subscribing to the IGMP multicast group will receive the periodic beacons.

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

The beacon message has the following format (field values are examples only):

```
AMXB
<-UUID=GlobalCache_000C1E5C0A5C>
<-SN=0000001>
<-SDKClass=Utility>
<-Make=GlobalCache>
<-Model=SL232>
<-Revision=710-4002-02>
<-Config-URL=http://192.168.1.127>
<-Status=Eth-Up,Host:OK>
<-Mod_Addr=1>
<-IO_Class=serial>
<-IO_ID=SL232>
<-PCB_PN=025-0042-13>
```

The Model field can be: SL232|RL3A|HMX3|IR3 or SL|RT|HM|IR(1)

(1)These model values are used on units produced Q4 2019 or later.

The UUID field value contains the Global Connect unique MAC address.

Global Connect also provides local name resolution using the multicast DNS mechanism, which also uses the UDP protocol. (See RFC 6762 for additional details, https://tools.ietf.org/html/rfc6762.) This mechanism makes the Global Connect discoverable using a network name of the format <networkName>, where <networkName> is the value assigned in the Name field of the Global Connect network settings.

4. API CONNECTION

All Global Connect TCP API requests and responses (except for serial data as noted below) are communicated through TCP socket connection on port 4998. Up to ten simultaneous TCP API client connections are supported. All TCP API requests are sent as a single line ASCII text string terminated with a carriage return. The requests follow a consistent format, which requires a command, and typically one or more associated parameters specifying module and connector/port. Often additional comma delimited parameters for command-specific options are required. See the Section 5 API Requests for detailed explanation.

All Global Connect modules support a common set of TCP API commands, while each unique module type supports commands specific to its function. See specific details in the following sections for each Global Connect module type.

Note that the serial module supports a unique function. Sending and receiving data to and from the serial port is accomplishing by a raw TCP connection on port 4999. In this case, data is communicated directly between the network TCP connection and the serial connector in a TCP-to-serial “bridge” mode. The data is not interpreted or altered by the device in any way. See Section 5.3 for additional details.

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

5. API REQUESTS

The structure of all TCP API requests for Global Connect and Global Connect modules are described in following section and subsections.

With few exceptions, TCP API requests adhere to the following format:

`<command>,<module>:<port>,<parameter1>,<parameter2>,…,<parameterN>↵`

In the above line, the following details are noteworthy:

- Each parameter is represented by a name enclosed in brackets (<>).
- Optional parameters are represented by a name enclosed by square brackets ([]).
- Each parameter is separated by a comma.
- Various commands require different numbers of parameters.
- The request must be a single line ending with a carriage return (↵).
- `<command>` is always a fixed text command string.
- `<module>:<port>` is sometimes a fixed value.
- `<parameterX>` always represents a user specified text value and must be replaced with a value as indicated for the specific command. When multiple choices are available for the parameter they will be displayed delimited by a vertical bar separator (|) character.

Note: Commands and parameters are case sensitive.

Example: The **get_NET** command is used to query network settings:

`get_NET,<module>:<port>`

where:
`<module>:<port>` = `0:1` (the Global Connect host network module address is 0:1)

So, the literal TCP API request sent to Global Connect is:

`get_NET,0:1↵`

Note: get_NET command is fully documented in Section 5.1.

Errors

An ERR response is returned when a TCP API request is invalid.

Example invalid request:

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

```
setstate,1:1,↵
```
(<state> parameter is missing - see Section 5.4)

Response:

```
ERR RO002↵
```
(Relay error code indicating an invalid state parameter)

Note: Error Code definitions are provided in the Error Codes table in Section 6.

5.1 GENERAL COMMANDS

getdevices

Query current Global Connect modules. The response is multi-line, and lists each module with its address and type, including the Host (0,0), and configured Global Connect module. A complete response ends with an endlistdevices line.

The purpose of the **getdevices** command is to determine the functionality present on the modules. This allows drivers to determine what types of control are available on each module.

Request:

```
getdevices↵
```
(query for modules and capabilities)

Response:

```
device,<module>,<ports> <Module type>↵
…
device,<module>,<ports> <Module type>↵
endlistdevices↵
```

where:

<type>	= MODULE	
<ports>	= 1|3|4|6	(Number of ports available on module)
<Module type>	= IR_OUT|SENSOR_DIGITAL|SERIAL_RS232|RELAY_SPST_3A|SWITCH_HDMI_3:1[(1)]	
<module>	= 1	

[(1)] Information after the underline or underscore (_) character is considered additional information and should only be used to fill informational fields within the software. Following this practice allows drivers written now to work with modules that may be released in the future that operate similary. For example, an RS485 module would have module type SERIAL_RS485. Because RS485 and RS232 use the same API commands, a driver written for the SERIAL_RS232 module type would work with an RS485 module without the need for a programmer to review and possibly modify the existing driver.

Following are several example responses:

```
device,0,0 MODULE↵   (Infrared Module response)
device,1,3 IR_OUT↵
```


Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

```
device,1,3 SENSOR_DIGITAL↵
endlistdevices↵
```

```
device,0,0 MODULE↵  (Serial Module response)
device,1,1 SERIAL_RS232
endlistdevices↵
```

```
device,0,0 MODULE↵  (HDMI Module response)
device,1,4 SWITCH_HDMI_3:1
endlistdevices↵
```

```
device,0,0 MODULE↵  (Relay Module response)
device,1,6 RELAY_SPST_3A
endlistdevices↵
```

getversion

Query Global Connect firmware version.

Request:

```
getversion↵
```

Response:

<version Number>↵

where:
<version Number> = 710-4001-xx|710-4002-xx|710-4003-xx|710-4004-xx

Example response from Global Connect:

```
710-4003-02↵
```

get_NET

Query current network settings. The response is a single comma delimited line.

Request:

```
get_NET,0:1↵
```

Response:

```
NET,0:1,<configlock>,<ipsetting>,<ipaddress>,<subnet>,<gateway>↵
```


Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

where:

<configlock>	= LOCKED|UNLOCKED
<ipsetting>	= DHCP|STATIC
<ipaddress>	= IP address
<subnet>	= subnet mask
<gateway>	= network gateway

5.2 INFRARED (IR)

Global Connect, when used with IR emitters or IR blaster cables, can output standard IR protocol signaling and control a wide range of IR controlled devices. The IR receiver cable can be used to receive IR codes at long ranges, and the built in IR learner can be used for learning IR codes at short distances. TCP API commands for IR functionality are detailed in the following subsections.

5.2.1 IR COMMANDS

set_IR

This command allows configuration of the IR Ports to match the selected IR cable.

Request:

```
set_IR,<module>:<port>,<mode>↵
```

where:

<module>	= 1
<port>	= 1|…|3
<mode>	= IR|IR_BLASTER|SENSOR|SENSOR_NOTIFY|RECEIVER

Note: Only IR port 3 supports IR_BLASTER.

Note: SENSOR and SENSOR_NOTIFY modes are further defined in section **5.6 Sensor Input**.

Response:

```
IR,<module>:<port>,<mode>↵
```

Example request and response:

```
set_IR,1:1,IR↵
```
Configures IR port 1 for a Global IR Emitter cable.

```
IR,1:1,IR↵
```

```
set_IR,1:3,IR_BLASTER↵
```
Configures IR port 3 for an IR blaster cable.

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

```
IR,1:3,IR_BLASTER↵
```

get_IR

Retrieve the current mode setting for the IR module.

Request:

```
get_IR,<module:port>↵
```

where:
<module> = 1
<port> = 1|…|3

Response:

```
IR,<module:port>,<mode>↵
```

where:
<mode> = IR|IR_BLASTER

5.2.2 IR STRUCTURE

An IR transmission is created by sending an IR timing pattern to Global Connect devices. The pattern is a series of <on> and <off> states modulated with a carrier frequency (ƒ) which is present only during the <on> state. A carrier frequency is typically between 35 to 45 KHz, with some equipment manufacturers using as high as 500 KHz. The length of time for an <on> or <off> state is calculated in units of the carrier frequency period. For example, an <off> value of 24 modulated at 40 KHz produces an <off> state of 600µS, as calculated below.

A period is 1 /ƒ or 1/40000 or .000025 seconds or 25µS,
and a value of 24 periods is 600µS

Figure 5.2.1 illustrates an IR timing pattern that would be created for the value shown below. IR timing patterns typically have a long final <off> value (or rest state) to ensure the next IR command is interpreted as a separate IR transmission.

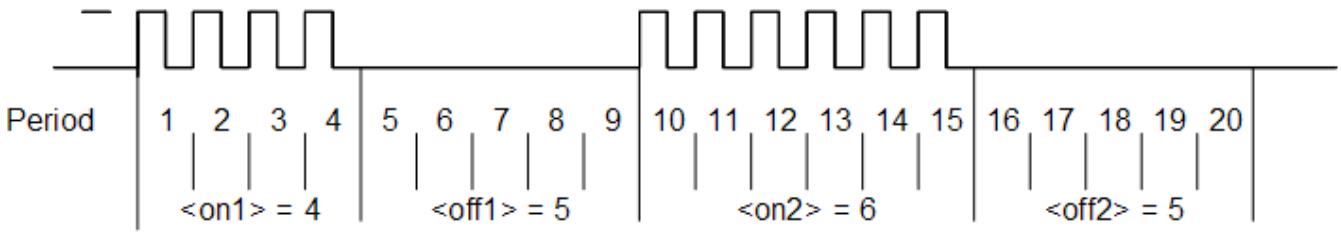


Figure 5.2.1

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

5.2.3 SENDING IR

Control of IR devices is accomplished through use of the `sendir` command. IR commands can take several hundred milliseconds to transmit, so Global Connect provides a `completeir` response to indicate when the device is ready to accept a new IR command.

sendir

Request:

```
sendir,<module:port>,<ID>,<freq>,<repeat>,<offset>,<on1>,<off1>,<on2>,<off2>,…,<onN>,<offN>↵
```

where:

<module:port> = [1] address of the desired IR channel (see below)
<ID> = 0|1|2|…|65535 [2] (for the completeir response, see below)
<freq> = 15000|…|500000 (in hertz)
<repeat> = 1|2|…|20 [3] (the IR command is sent <repeat> times)
<offset> = 1|3|5|…|383 [4] (used if <repeat> is greater than 1, see below)
<on1> = 4|5|…|50000 [5] (number of pulses)
<off1> = 4|5|…|50000 [5] (absence of pulse periods of the carrier frequency)
N = the count of <on>,<off> pairs, which must be less than 260 (total of 520 numbers)

[1] Global Connect IR3 provides three (3) discrete channels of IR output. In these cases, sendir requests to any of the 3 IR channels must use the port component of the <module:port> parameter to address each of the 3 IR channels. Thus, for Global Connect, each of the 3 IR channels is addressed as follows (not all sendir parameters are shown):

```
sendir,1:1,…
sendir,1:2,…
sendir,1:3,…
```

[2] <ID> is a number provided by the sendir request, which is later included in the completeir response to indicate successful completion of the requested IR transmission.

[3] <repeat> is the number of times an IR transmission is sent, if not aborted via a stopir, or subsequent IR command (see section 5.4). Values above 20 are accepted, but the maximum number of transmitted repeats is capped at 20. In all cases, the preamble is only sent once (see <offset> below).

[4] An <offset> applies when the <repeat> is greater than 1. For IR commands that have preambles, an <offset> is employed to avoid repeating the preamble during repeated IR timing patterns. The <offset> value indicates the location within the timing pattern to start repeating the IR command as indicated below. The <offset> will always be an odd value since a timing pattern begins with an <on> state and must end with an <off> state.

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

<offset> odd value	Repeat start locations	
1	<on1>	no preamble
3	<on2>	
….	….	
n-1	<on((n/2)-1)>	

(5) Since IR transmissions end in an <off> condition, there must be an equal number of <on> and <off> states. Also, every <on> and <off> state must meet an 80µS minimum time requirement for the IR codes to be sent properly.

Example: With a carrier frequency of 60 KHz, the minimum value for <on> and <off> states is calculated below.

$$<off>_{min} = <on>_{min} \geq 80\mu S * f = 80\mu S * 60\ KHz = 4.8$$

For precise replication of an IR code at 60 KHz, all <on> and <off> values in the timing pattern must be 5 or higher.

All of the conditions above must be met for a valid sendir request. When a variable is missing or outside the accepted range, an error will be returned. For example, the sample sendir commands below will result in an error response.

Request:
`sendir,10:3,3456,23400,1,1,24,48,24,960`↵ (invalid module number for Global Connect)
Response:
`ERR 003`↵

Request:
`sendir,1:1,23333,40000,2,3,24,48,24,48,960`↵ (not an equal number of <on> and <off> parameters)
Response:
`ERR IR006`

Request:
`sendir,1:1,0,40000,2,2,24,48,24,960`↵ (<offset> is an even number)
Response:
`ERR IR004`↵

GLOBAL CACHÉ COMPRESSED IR FORMAT

The compressed IR format interprets the first 15 unique <on>,<off> pairs and assigns a single uppercase letter (A,B,C, and on) to each of those unique pairs. The first time a unique pair occurs within an IR command it is assigned a single ASCII character. Any subsequent occurrences of that same pair are then represented with that same ASCII character instead of the <on>,<off> representation. This allows the sendir request string to be significantly shorter, which can be advantageous in the case of a long IR code.

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

Example:

```
sendir,1:1,2445,40000,1,1,4,5,4,5,8,9,4,5,8,9,8,9↵
```

The unique pairs are underlined in the example request above. By assigning A to 4,5 and B to 8,9, the request can be rewritten as follows:

```
sendir,1:1,2445,40000,1,1,4,5A8,9ABB↵
```

Both commands are syntactically correct and are accepted by Global Connect. Both will transmit an identical IR code.

completeir

After the Global Connect successfully processes and transmits an IR command, it responds by sending the completeir response to the requester. The completeir response can be associated with the originating sendir command by way of the <ID> parameter. When utilized, the <ID> can provide a unique identifier to determine which IR transmissions have completed.

Sent from Global Connect in response to successful sendir:

```
completeir,<module>:<port>,<ID>↵
```

where:

<module>	= 1	
<port>	= 1|…|3	
<ID>	= 0|1|2|…|65535	(value is generated by the sender of the sendir request)

Examples:

Request:

```
sendir,1:1,2445,40000,1,1,4,5,6,5↵
```

Response:

```
completeir,1:1,2445↵
```

The following examples demonstrates two commands to send the same simple IR timing pattern of 24,12,24,960, repeated four times, and with a preamble of 34,48:

```
sendir,1:1,4444,34500,1,1,34,48,24,12,24,960,24,12,24,960,24,12,24,960,24,12,24,960↵
sendir,1:1,34,34500,4,3,34,48,24,12,24,960↵
```

Responses to the above requests:

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

```
completeir,1:1,4444↵
completeir,1:1,34↵
```

The same IR command is repeated four (4) times by either sendir request, but since the <ID> is different, a different completeir response is returned. The <ID> also allows associating each request to its response. Note that the second IR request is the recommended format, since it is shorter and also allows the command to be halted in between repeats, if necessary

stopir

This request will stop an active IR transmission. Any remaining <repeat> cycles will be aborted. A stopir command sent to a connector not configured as an IR output will return an error message. An IR transmission halted with the stopir command will return a stopir response. Furthermore, if an IR command is halted before its completion by another connection, the originating IR connection and the connection sending stopir will both receive a stopir response. If stopped, the originating connection will not receive a completeir response.

Request:

```
stopir,<module>:<port>↵
```

Response:

```
stopir,<module>:<port>↵
```

where:
<module> = 1
<port> = 1|…|3
A stopir command always returns a stopir response, regardless of whether the port is idle, or an IR active transmission is halted. A stopir response means only that the stopir command was successfully sent to Global Connect and any transmission has been halted.

busyIR

A busyIR response occurs when an IR command is received for a port which is actively transmitting an IR code. This might occur, for example, if multiple IP connections are present (such as from multiple network users). In this case, the sendir command that receives the busyIR response will have its IR code transmitted.

Response to an attempt to interrupt IR transmission by another IP connection or socket:

```
busyIR,1:1,<ID>↵
```

where:
<ID> = 0|…|65535 (ID is specified in sendir command)

Note: The busyIR response is sent to the originator of the failed IR command. A stopir command will only return a stopir response. A stopir command will never return a busyIR response.

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

5.2.4 SMOOTH CONTINUOUS IR COMMANDS

A general discussion is necessary to better understand how smooth continuous IR commands are executed by Global Connect. This feature is utilized for smooth volume control or repeating an operation without the appearance of choppy actions. The approach of sending an IR command with large repeat counts and stopping it upon request will work but can lead to undesirable incidents. Consider an IR command with a large repeat count used to increase volume in a smooth fashion. The IR command continues repeating while volume continuously increases. But if the controlling client/application unexpectedly disconnects, the volume could continually increase (possibly damaging equipment) until the client reconnects and issues a stopir command.

Global Connect's solution is to limit the repeat count. To create smooth IR operation, Global Connect resets the IR repeat count each time the identical IR command (from the same IP connection) is resent. This method will not interrupt and restart the IR command, but reset the IR repeat count back to the original value.

Example: If the IR repeat count is set to 5, and the IR command has transmitted 3 times, receipt of the same command causes the repeat count to be reset back to 5. This process can continue indefinitely while a volume button is held down to create smooth operation. However, at no time can the command repeat more than 5 times after the button is released or an IP connection is inadvertently lost, preventing a potentially serious issue.

By selecting an appropriate <repeat> value, the need for a stopir command is eliminated. In this example the volume continues to increase smoothly by retransmitting repeated IR commands due to the volume button being pressed. If the next repeated IR command is received before the previous command finishes, smooth operation is realized. By choosing a low repeat value, the volume increase will stop when the volume button is released. Also, proper IR operations happen even with unintended network delays due to traffic or WiFi connectivity. In this unlikely event, only small hesitations will be experienced during IR operation.

If the identical command is not received before the original command is finished, the command will be registered as a brand new command and is sent as such. The command in question will operate functionally the same, but delays between commands may be evident when used in this way. Increasing the <repeat> value will likely eliminate these discrepancies.

5.2.5 RECEIVING IR

The Global Connect can receive IR codes through both the IR module's built in IR learner as well as using an IR receiver cable. The IR learner is designed for learning IR codes, while the IR receiver is a long-range receiver and is designed to be used by control applications either as a control input or for performing IR "tunneling" over the network.

receiveIR

Start or stop IR receiver mode for use with the IR receiver cable. Cable type must be set to RECEIVER using the **`set_IR`** command, or through the web interface or HTTP API.

Once IR receiver mode is enabled, any IR signals received by the receiver cable will be sent to the connected TCP client that enabled the mode, until IR receiver mode is disabled, or the TCP connection is terminated.

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

Received IR codes will be in the uncompressed Global Caché command format as described in the sendir command in **section 5.2.3** with a module and port value of 1.

Request:

receiveIR,<module>,<port>,<mode>

where:
<module> = 1
<port> = 1[(1)]
<mode> = start|stop

Response:

receiveIR,<module>,<port>,<state>

where:
<module> = 1
<port> = 1|2|3
<state> = enabled|disabled

(1): The port value of the received IR code will be changed in a future update to match the IR port number that the IR code was received on. This will allow for differentiating between IR ports when multiple receiver cables are being used on the same IR module.

Example:

Request:

```
receiveIR,1:1,enabled↵
```

Response:

```
receiveIR,1:1,enabled↵

sendir,1:1,1,36429,1,1,95,34,15,17,15,17,15,34,15,33,47,34,15,17,15,17,15,17,15,2521↵
```

get_IRL

The get_IRL command activates the built in IR learner. Once activated IR codes received by the learner will be returned to the TCP client that sent the command. IR learner can be disabled by closing the TCP connection or sending the **stop_IRL** command.

Received IR codes will be in the uncompressed Global Caché command format as described in the sendir command in **section 5.2.3** with a module and port value of 1.

Example:

Request:

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

```
get_IRL↵
```

Response:

```
IR Learner Enabled↵

sendir,1:1,1,36429,1,1,95,34,15,17,15,17,15,34,15,33,47,34,15,17,15,17,15,17,15,2521↵
```

stop_IRL

The stop_IRL command deactivates learner mode.

Example:

Request:

```
stop_IRL↵
```

Response:

```
IR Learner Disabled↵
```

5.3 SERIAL

The SL232 serial module allows for bi-directional serial communications over a RS232 interface. Serial communication parameters are configured using Global Connect embedded web pages or by direct API commands.

To send and receive serial data, a TCP socket connection must be established on port 4999. Data is communicated directly between the TCP connection and the serial connector in a TCP-to-serial bridge mode. The data is passed directly and is not interpreted or altered in any way by Global Connect.

If Global Connect serial communication parameters are not configured correctly to match the connected device, buffer overflows (indicating data loss) or parity errors may occur.

5.3.1 MULTIPLE SERIAL CONNECTIONS

The Global Connect supports up to four (4) simultaneous, bidirectional TCP-to-serial connections on port 4999.

SENDING DATA TO A SERIAL DEVICE

In order to support multiple TCP clients simultaneously sending data to a single serial-connected device, Global Connect handles received TCP data on a packet basis for efficient transmission to a serial-connected device. When a data packet is received by the Global Connect it is transmitted completely to the serial-connected device before any subsequent packets (received from the same or different socket) are sent. This is important when considering the case where TCP client A sends a single command in two

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

(2) separate packets while a different TCP client B simultaneously sends a packet. In this case, the packet from TCP client B may be serviced in between the 2 separate packets received from TCP client A, possibly interrupting the data or command from TCP client A. It is very important for TCP clients sending data to the Global Connect to send a complete serial data sequence or command(s) within a single packet. This ensures complete data sequences or commands are received properly by the serially-connected device.

RECEIVING DATA FROM A SERIAL DEVICE

Serial data received by Global Connect from a serial-connected device is also handled as packets. But unlike TCP-to-Serial data flow, wherein packets are based on TCP protocol, serially received data is packetized according to timing and size. The initial character of received serial data initiates a receive packet. The completion of the packet occurs when either the Global Connect receive buffer reaches a certain capacity or no characters are received for a calculated time delay (based on current baud rate). The completed serial data packet is then immediately transmitted to all active/open TCP socket connections. This allows all connected TCP clients to synchronously receive all data sent from the serial device.

5.3.2 SERIAL COMMANDS

set_SERIAL

Configure Global Connect serial communication settings.

Requests:

```
set_SERIAL,<module>:<port>↵

set_SERIAL,<module>:<port>,<baudrate>,<flowcontrol>,<parity>,[stopbits]↵
```

Response:

```
SERIAL,1:1,<baudrate>,<flowcontrol>,<parity>,<stopbits>↵
```

where:
<module>:<port>= 1:1
<baudrate> = 300|…|115200
<flowcontrol> = FLOW_HARDWARE|FLOW_NONE

<parity> = PARITY_NO|PARITY_ODD|PARITY_EVEN
[stopbits] = STOPBITS_1|STOPBITS_2 (optional parameter)

Example request and response:

```
set_SERIAL,1:1,38400,FLOW_NONE,PARITY_NO↵

SERIAL,1:1,115200,FLOW_NONE,PARITY_NO,STOPBITS_1↵
```


Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

get_SERIAL

Query current serial communications settings.

Request:

```
get_SERIAL,<module>:<port>↵
```

Response:

```
SERIAL,1:1,<baudrate>,<flowcontrol>,<parity>,<stopbits>↵
```

where:
<module>:<port>= 1:1
<baudrate> = 300|…|115200
<flowcontrol> = FLOW_HARDWARE|FLOW_NONE

<parity> = PARITY_NO|PARITY_ODD|PARITY_EVEN
<stopbits> = STOPBITS_1|STOPBITS_2

Example request and response:

```
get_SERIAL,1:1↵

SERIAL,1:1,115200,FLOW_NONE,PARITY_NO,STOPBITS_1↵
```

5.4 RELAYS

The GCRL3A relay module provides dry contact closure relay outputs capable of switching a variety of devices. See the Global Connect data sheet for detailed hardware specification.

This section describes the TCP API commands used for configuration and control of the relay outputs.

5.4.1 RELAY COMMANDS

get_RELAY
Query the type of a specified relay port.

Request:

```
get_RELAY,<module>:<port>↵
```


Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

where:
<module> = 1
<port> = 1|…|6

Response:

```
RELAY,<module>:<port>,<type>↵
```

where:
<type> = SPST

Example request and response:

```
get_RELAY,1:4↵

RELAY,1:4,SPST↵
```

set_RELAY

Set the type of a specified logical relay port.

Note: This command is for backwards compatibility with the Flex Link Relay and Sensor Cable. It provides no functional purpose otherwise.

Request:

```
set_RELAY,<module>:<port>,<type>↵
```

where:
<module> = 1
<port> = 1|…|6
<type>[1]SPST

Response:

```
RELAY,<module>:<port>,SPST↵
```

Example request and response:

```
get_RELAY,1:3↵                    Query the current configuration for logical relay port 3.
RELAY,1:3,SPST↵
```

getstate

Query the current state of a logical relay port.

Request:

```
getstate,<module>:<port>↵
```


Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

where:
<module> = 1 [1]
<port> = 1|…|6

Response:

```
state,<module>:<port>,<state>↵
```

where:
<state>
0 = off (open)
1 = on (closed)

[1] A module value of 3 is also accepted for GC-100 backwards compatibility.

setstate

Set the state of a logical relay port.

Request:

```
setstate,<module>:<port>,<state>↵
```

where:
<module> = 1 [1]
<port> = 1|…|6
<state>
0 = off (open)
1 = on (closed)

Response:

```
state,<module>:<port>,<state>↵
```

Note: Relay states are not preserved if the Global Connect module is unplugged or reinitialized or if Global Connect is reset and/or power cycled. Under any of those conditions, all relays will return to their default inactive (open) state.

[1] A module value of 3 is also accepted for GC-100 backwards compatibility.

5.5 HDMI SWITCHING/CEC

The GCHMX3 module is a 3-input 1-output HDMI switch that supports UHD 4K 60Hz. The TCP API allows control of the HDMI switch ports, determining which ports are active, and sending and receiving CEC commands.

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

5.5.1 HDMI SWITCHING COMMANDS

setstate

Select the active HDMI switch input and output.

Request:

```
setstate,<module>:<in_port>,<out_port>↵
```

where:

<module> = 1

<in_port> =

1|…|4

0 = disable specified <out_port>

<out_port>

1

0 = disable specified <in_port>

Response:

```
state,<module>:<port>,<state>↵
```

Enabling an input will automatically disable any other enabled inputs and will enable the output if it is not enabled. Disabling an active input will also disable the output.*

***Important Note:** Currently disabling an inactive input also disables the output. This behavior is unintended and will be updated in a future update.

Example request and response:

```
setstate,1:1,1↵      (Switch the input 1 to the output)
state,1:1,1↵

setstate,1:3,0↵      (Disable input 3 if enabled)
state,1:3,0↵
```

getstate

Returns the current selection state of a HDMI input and whether it is connected to an output.

Request:

```
getstate,<module>:<in_port>,[notify]↵
```


Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

where:
<module> = 1
<in_port> = 1|…|4
[notify] = notify (optional parameter)

The optional [notify] parameter enables TCP notifications of state information. TCP clients that send the notify command for a sensor input will receive an immediate state response and then state responses from that input (on state changes) until the TCP connection is terminated. Commands that do not include the notify command will receive the current state of the input and no subsequent updates.

Example request and response without change-notification:

```
getstate,1:2↵
state,1:2,1↵ (Input 2 is enabled)
```

Example request and response with change-notification:

```
getstate,1:2,notify↵
state,1:2,1↵ (Input 2 is enabled)
state,1:2,0↵ (Input 2 is disabled)
state,1:2,1↵ (Input 2 is enabled again)
```

5.5.2 HDMI PORT STATUS COMMANDS

getactive

Returns the state of the connected HDMI devices. A true state value indicates that an HDMI cable is connected to the port, and that the connected source/sink device is powered on.

Request:

```
getactive,<module>↵
```

where:
<module> = 1

Response:

```
active,<module>↵
OUT:
1,<state>
IN:
1,<state>↵
2,<state>↵
3,<state>↵
```


Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

```
endactive,<module>↵
```

where:
<module> = 1
<port> = true|false

Example request and response:

```
getactive,1↵

active,1↵
OUT:↵
1,true↵
IN:↵
1,true↵
2,false↵
3,false↵
endactive,1↵
```

5.5.3 CEC COMMANDS

CEC

The CEC command allows for transmission and reception of CEC commands. Transmission of CEC commands (Consumer Electronics Control) allows applications to control CEC enabled equipment as well as query state information. The CEC reception functionality allows for receiving of all CEC messages transmitted on the CEC bus. This can be used to monitor CEC commands being sent and received by other devices or to see feedback from transmitted query commands.

Request:

```
CEC,<module>,<out_port>,<mode>,<value>↵
```

where:

<module> = 1
<out_port> = 1
<mode> = TX|RX
<value> = if <mode> = TX ... colon-delimited CEC message bytes
if <mode> = RX ... enabled|disabled|on|off|1|0

Response:

```
CEC,<module>,<out_port>,<mode>,<value><,ACK value>↵
```

where:

<module> = 1
<out_port> = 1
<mode> = TX|RX

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

<value>	=	if <mode> = TX ... colon-delimited CEC message bytes if <mode> = RX ... enabled|disabled|on|off|1|0
<,ACK value>	=	,ACK (unicast CEC message was acknowledged by the destination) ,!ACK (unicast CEC message was not acknowledged by the destination) ,NACK (broadcast CEC message was rejected by a device) , (broadcast CEC message was not rejected by any devices)*

***Note**: This trailing comma is unintended and will be removed in a future firmware update. Received broadcast messages not explicitly rejected include no <,ACK value> parameter. Driver implementations should take this in account when implementing CEC functionality using this version of the API.

Unicast CEC messages should be acknowledged (ACK'd) by the destination device. If no ACK occurs, ",!ACK" will be appended to the response. If no ACK is received, 2 retries will occur (for a total of 3 transmit attempts), until finally returning error code SW011 after the final (3rd) unacknowledged transmit.

Conversely, broadcast messages (sent to logical address "F") need not be ACK'd by receiving devices, but can be NACK'd (rejected) by any device. NACK'd messages will have ",NACK" appended to the response. If no NACK is received, the broadcast message was successful.

When the CEC RX mode is enabled, any CEC messaged transmitted (by any CEC device) will be sent as a response to the connected TCP client. This will continue until CEC RX mode is disabled or the TCP connection is terminated.

Example request and response:

`CEC,1:1,TX,40:04↵` (Playback 1 to TV – Image View On ... turns display on)

`CEC,1:1,TX,40:04,ACK↵`

Example response <mode>=RX:

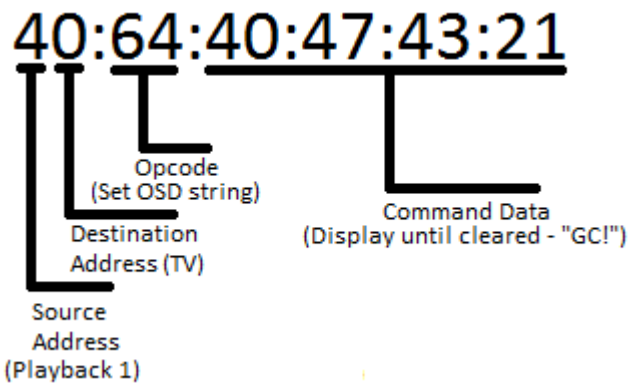
`CEC,1:1,RX,0A,ACK↵` (TV to Tuner 4 - Play)

5.5.4 CEC COLON-DELIMITED COMMAND FORMAT

The <value> parameter described in Section 5.5.3 represents the actual CEC message in a transmit or receive CEC command. This CEC message is represented as colon-delimited hexadecimal values. The first byte of the command indicates the logical address of the transmitting and receiving devices. The second byte indicates the opcode. The remaining bytes, if necessitated by the opcode, are the message data.

Example CEC command:

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800



For a full list of command opcodes and their function please see the latest CEC specification document.

The CEC logical addresses are listed below. These addresses are used in the source address and destination address portions of the command. The logical addresses are represented by hexadecimal characters. CEC devices will negotiate a logical address in a process described in the CEC specification.

Address	**Device**
0 (0x0)	TV
1 (0x1)	Recording Device 1
2 (0x2)	Recording Device 2
3 (0x3)	Tuner 1
4 (0x4)	Playback Device 1
5 (0x5)	Audio System
6 (0x6)	Tuner 2
7 (0x7)	Tuner 3
8 (0x8)	Playback Device 2

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

9 (0x9)	Recording Device 3
10 (0xA)	Tuner 4
11 (0xB)	Playback Device 3
12 (0xC)	Reserved
13 (0xD)	Reserved
14 (0xE)	Free Use
15 (0xF)	Unregistered (as source address) Broadcast (as destination address)

5.6 SENSOR INPUT

Global Connect sensor inputs allow sensing of contact closure, and presence of voltage. Several adapter cables are available depending on which type of input is being sensed, as follows:

GC-SC1 (contact closure)
GC-SP1 (voltage)
GC-SV1 (RCA video)

Before sensor input commands can be used, the port must be configured for sensor input using the **set_IR** command in **section 5.2.1**.

5.6.1 SENSOR INPUT COMMANDS

getstate

The **getstate** command allows for querying the current state of the sensor cable.

Request:

```
getstate,<module>:<port>,[notify]↵
```

where:
<module> = 1

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

<port> = 1|2|3
[notify] = notify

Response:

```
state,<module>:<port>,<state>↵
```

where:
<module> = 1
<port> = 1|2|3
<state> = 0|1

The optional [notify] parameter enables TCP notifications of state information. TCP clients that send the notify command for a sensor input will receive an immediate state response and then state responses from that input (on state changes) until the TCP connection is terminated. Commands that do not include the notify command will receive the current state of the sensor cable and no subsequent updates.

Example request and response without change-notification:

```
getstate,1:1↵

state,1:1,0↵
```

Example request and response with change-notification:

```
getstate,1:2,notify↵

state,1:2,1↵
state,1:2,0↵
state,1:2,1↵
```

set_SENSORNOTIFY

The **set_SENSORNOTIFY** command sets configuration values pertinent to sensor notify modes (both TCP and UDP). This command also can be used to enable the UDP sensor notify feature, which provides automatic/asynchronous notification of changes in sensor input state. When enabled, a UDP multicast message can be broadcast when either of the following conditions occurs:

- A time interval elapses.
- A sensor input changes state.

Both the time-interval and UDP port-number are configurable for each of the sensor ports. The debounce value can also be set on a port basis. State changes that last less than the debounce value will result in the temporary state not being reported via notification methods, however a notification will be sent with the final resting state that fulfills the debounce requirements.

Request:

```
set_SENSORNOTIFY,<module>:<port>,<notify_port>,<notify_interval>,[debounce]↵
```


Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

where:

<module>	= 2	
<port>	= 1|…|4	
<notify_port>	= 0|…|65535	('0' = all beacons disabled)
<notify_interval>	= 0|…|65535	('0' = periodic beacon disabled)
[debounce]	= 10us|…|1s	(optional parameter)

Response:

```
SENSORNOTIFY,<module>:<port>,<notify_port>,<notify_interval>,<debounce>↵
```

The default values are <notify_port> = 0 (which disables both the state-change and periodic notification beacons), and <notify_interval> = 0 (which disables only the periodic notification beacon), and [debounce] = 100ms (which filters state changes less than 100mS in duration). The disabled-by-default notify settings avoid a potential flood of undesired network UDP traffic in default configuration and allow custom user configuration according to the specific network environment.

When selecting UDP port values, it is advisable to avoid conflicts with any ports already in use by the network environment. Please consider all connected network hardware and refer to various available standards registries (such as the IANA Service Name and Transport Protocol Port Number Registry, https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt) for a list of assigned vs. available port numbers. For example, according to the IANA registry, UDP ports 9132 - 9159 are unassigned and could be a good choice if not already used by other network devices.

Example request and response 1:

```
set_SENSORNOTIFY,1:1,9132,10↵
SENSORNOTIFY,1:1,9132,10,100ms↵
```

Example request and response 2:

```
set_SENSORNOTIFY,1:1,9132,10,500ms↵
SENSORNOTIFY,1:1,9132,10,500ms↵
```

6. ERROR CODES

The table below lists of error messages returned by Global Connect in response to various invalid TCP API requests. See section 5 for an example invalid request and resulting response format.

General Errors	Explanation
ERR 001	Invalid request. Command not found.
ERR 002	Bad request syntax used with a known command.

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

ERR 003	Invalid or missing module and/or connector address.
ERR 004	No carriage return found.
ERR 005	Command not supported by current port setting.
ERR 006	Settings are locked.
ERR 007	Internal Error.
IR Errors	**Explanation**
ERR IR001	Invalid ID value.
ERR IR002	Invalid frequency.
ERR IR003	Invalid repeat.
ERR IR004	Invalid offset.
ERR IR005	Invalid IR pulse value.
ERR IR006	Odd amount of IR pulse values (must be even).
ERR IR007	Maximum pulse pair limit exceeded.
ERR IR008	Port unavailable/busy.
ERR IR020	IR input busy.
ERR IR021	IR input data overflow.
Sensor Errors	**Explanation**
ERR SI001	Sensor not enabled.

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800

ERR SI002	Invalid port value.
ERR SI003	Invalid timer value.
Serial Errors	**Explanation**
ERR SL001	Invalid baud rate.
ERR SL002	Invalid flow control setting.
ERR SL003	Invalid parity setting.
ERR SL004	Invalid stop bits setting.
Relay	**Explanation**
ERR RO001	Invalid logical relay type.
ERR RO002	Invalid logical relay state.
ERR RO003	Unsupported operation.
ERR RO004	Logical relay disabled or not available.
HDMI	**Explanation**
ERR SW001	Invalid state value.
ERR SW010	Frame too large.
ERR SW011	CEC command was not acknowledged.

Global Cache'
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530www.globalcache.com
Phone: 541-899-4800