Global Caché Unified TCP API
Version 1.1

Table of Contents

1. Introduction 2

2. Network Connection 2

3. API Connection 2

4. API Requests 3

4.1 Command/Response Format 3

4.2 Unified Command Specification 4

4.2.1 Example 5

4.3 Common Commands 6

4.4 Class-Specific Commands 10

4.4.1 Infrared (IR) 10

4.4.2 Serial 16

4.4.3 Relays 18

4.4.4 Sensors 20

4.4.5 Matrix/Switch 22

Appendix A – Infrared Signaling and Codes 28

Appendix B – Serial To Network Bridging 31

Appendix C – Change Notification 32

Appendix D – Configurable Relays 33

5. API Errors 38

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

1. INTRODUCTION

The Global Caché TCP API provides a simple yet powerful interface to the entire family of Global Caché products which are based on a concept of network-connected devices having addressable modules and ports organized according by their functional class. Each functional class has a common set of API commands for implementing its functionality, and these API commands are compatible with any device in that class from any product family.

Our modular design approach provides significant benefits in flexibility and portability for API users. A single driver can be written using common (to all products) API commands to discover Global Caché devices on a network and determine each device's available modules and ports, as well as their functional classes. Because class-specific commands are compatible across all products, a single driver can be designed to work across all products.

This TCP API specification unifies and updates all information formerly provided in separate documents for each product line and concisely presents one resource for all API information required by driver and application developers.

2. NETWORK CONNECTION

Global Caché network-connected products currently support network connectivity through Ethernet or WiFi, depending on product line and model. By default, all devices are configured to automatically acquire their IP configuration via DHCP. However, if a DHCP server is not available, devices will assume a default static IP address as specified in each device's documentation.

Configuration of a device's network and I/O settings can be managed through the product's configuration web pages, or via API commands if supported by the product (see **get_NET** and **set_NET**).

For detailed information about configuration and operation of the various products from a user perspective, please refer to each product's Quick Start or User Guide.

3. API CONNECTION

TCP API requests and responses are sent and received over raw TCP socket connection to port number 4998. Socket connections can be momentary (send request, receive response, immediately disconnect) or persistent (keep a connection open for multiple requests and/or responses). Each product line supports a maximum number of simultaneous open connections, as shown below.

Maximum Simultaneous TCP Connections			
GC-100	iTach	Flex	Global Connect
1	8	8	8

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

4. API REQUESTS

4.1 COMMAND/RESPONSE FORMAT

TCP API requests follow a simple serialized command/response pattern wherein a command is sent by a network client to the device and a response is sent by the device back to the network client.

Requests follow a consistent format comprised of a single line of printable text starting with a command name often followed by comma-delimited parameter(s). Responses usually follow the same format, except in several cases where the response includes multiple lines. Requests must end with a carriage-return, and responses always end with a carriage-return.

In the scope of this document, the format of a command and associated parameters is always specified as shown below.

command,*<module>*:*<port>*,*<parameter1>*,*<parameter2>*,…,*<parameterN>*,*[parameter]*

Notes:

- A complete TCP API request consists of a single line of printable text ending with a carriage-return (ASCII value 13).
- Commands and parameters are case sensitive.
- A request begins with a unique command name, followed by a comma (if parameters are present) or carriage-return.
- Required parameters are represented by a unique string enclosed in angle brackets (**<>**).
- Optional parameters are represented by a unique string enclosed in square brackets (**[ ]**).
- Consecutive parameters are delimited by a comma (**,**).
- Parameter values are specified by the client (in the request) or returned by the device (in the response). In either case, a parameter is always limited to a set of valid values. In this document, valid values are specified by explicit list, or by range. In both cases a vertical bar is used in the notation as follows:
  - In an explicit list to delimit values (for example, **1**|**3**|**7**).
  - In a range… to separate min. and max. values, with an ellipsis representing intermediate integer values (for example, **0**|…|**9**).

The response to a request indicates success or error. The response may also return information such as settings or status.

The format of a success response usually echoes the command and parameters, as shown below. Note that the response name typically excludes the command's verb prefix, for example, the **get_NET** command which responds with **NET**.

response,*<module>*:*<port>*,*<parameter1>*,*<parameter2>*,…,*<parameterN>*,*[parameter]*

The format of an error response varies across different product lines, but generally follows the format shown below.

<error_prefix><error_code>

See the API Errors section for a complete list of error codes for each product line.

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

4.2 UNIFIED COMMAND SPECIFICATION

The TCP API is designed to be simple and consistent across all Global Caché product lines even though they each have unique properties and capabilities. This provides application and driver developers the significant advantage of reliable compatibility and reduced complexity across all products, thereby minimizing development and maintenance efforts.

This document presents a unified specification for TCP API commands for all Global Caché products. Unique or closely related commands are presented in their own sections. Each section's heading shows the command name(s) with a brief description, followed by one or more paragraphs describing the purpose, function, and usage. Then, a command compatibility matrix specifies the command/response, format, and parameters, as well as applicability, options, and other considerations relative to each Global Caché product line.

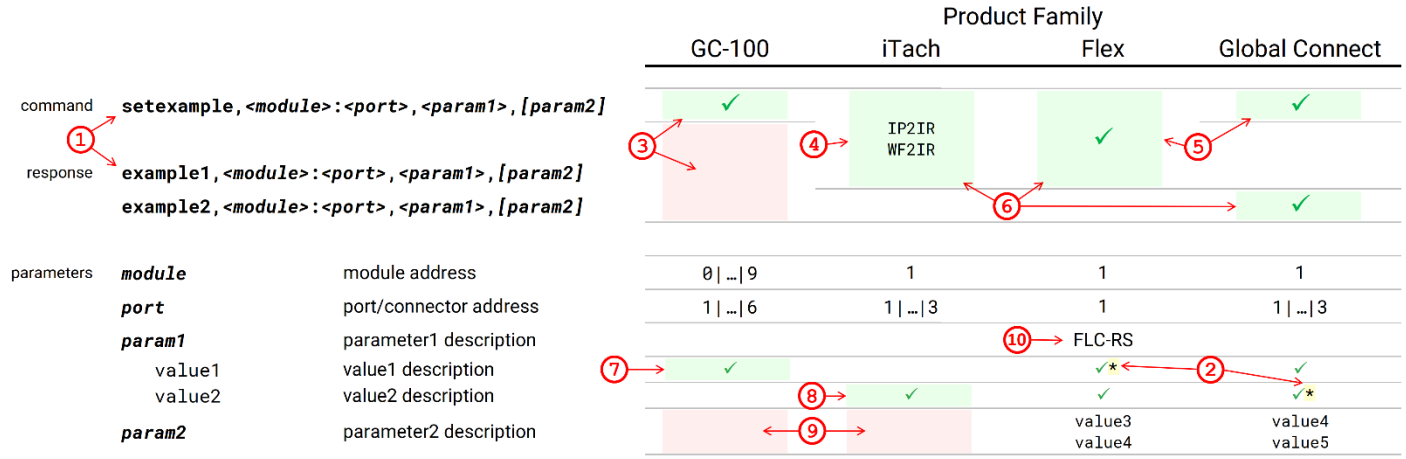
The following subsection provides an example command specification section for a fictional command **`setexample`**. Before proceeding to the actual command specifications, please review this example to gain an understanding of how API specification information is presented in this document. Important details and aspects of the command specification are highlighted and numbered with corresponding comments below the example section.

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

4.2.1 EXAMPLE

setexample – brief description

The **setexample** command performs an important function under certain conditions.



examples

GC-100-12

```
setexample,3:4,value1,valueX
```

iTach IP2IR

```
setexample,1:2,value2
example1,1:2,value2
```

Flex

```
setexample,1:1,value1,value3
example1,1:2,value1,value3
```

Global Connect

```
setexample,1:3,value2,value5
example2,1:2,value2,value5
```

1. Commands and responses are always a single line, ending with a carriage-return (unless specified otherwise). Some commands may be shown on multiple lines due to length.
2. Some parameters have factory-default values. These are indicated in the table by an asterisk (*). (Refer to each product's User Guide or Quick Start Guide for details on resetting to factory-defaults.)
3. All GC-100 models support this command, but no response is returned.
4. For iTach, only the IP2IR and WF2IR models support the command and the **example1** response.
5. All Flex and Global Connect models support this command.
6. iTach and Flex implement the **example1** response, while Global Connect implements **example2**.
7. For GC-100, only value1 is supported for ***param1***.
8. For iTach, only value2 is supported for ***param1***.
9. For GC-100 and iTach, ***param2*** is not supported.
10. For Flex, ***param1*** is used only for the FLC-RS Cable.

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

4.3 COMMON COMMANDS

This section covers TCP API commands for functions common to all product lines, models, and module classes, including device version, device capabilities, LED control, and network configuration.

getversion - get device version

The **getversion** command retrieves the device firmware version.

			Product Family			
			GC-100	iTach	Flex	Global Connect
command	**getversion[,*module*]**		✓	✓	✓	✓
response	**version,<*module*>,<*version*>**		✓			
response	**<*version*>**			✓	✓	✓
parameters	***module***	module address	0 | … | 9			
	version	version number	3.2-06 (-06) 3.2-12 (-12)	710-1005-XX (IP2IR) 710-1009-XX (IP2SL) 710-1008-XX (IP2CC) 710-1001-XX (WF2IR) 710-1007-XX (WF2SL) 710-1010-XX (WF2CC)	710-2000-XX (-WF) 710-3000-XX (-IP)	710-4001-XX (IR) 710-4002-XX (SL) 710-4003-XX (RL) 710-4004-XX (SW)

examples

GC-100-12

```
getversion↵
version,0,3.2-12↵
```

Flex WiFi

```
getversion↵
710-2000-20↵
```


Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

getdevices - get device capabilities

The **getdevices** command is used to enumerate device capabilities. The response contains multiple lines, one line for each module showing the module address, number of ports, and port type. This should provide enough information for applications to determine the appropriate driver(s). Newer devices may also include a port subtype parameter in the response, which provides additional information about the port type, but does not affect the driver determination.

The response always ends with a line containing only **endlistdevices**.

			Product Family			
			GC-100	iTach	Flex	Global Connect
command	**getdevices**		✓	✓	✓	✓
response	**device,*<module>*,*<ports>*␣*<type>*[*_subtype*]** **…** **endlistdevices**		✓	✓	✓	✓
parameters	***module***	module address	1 | … | 5	0 | … | 1	1[1]	0 | … | 1
	ports	port count	1 | … | 3	0 | … | 3	1[2]	0 | … | 6
	type	port type				
	ETHERNET			✓	✓	
	WIFI			✓	✓	
	MODULE					✓
	IR		✓	✓	✓	✓
	SERIAL		✓	✓	✓	✓
	RELAY		✓	✓		✓
	SENSOR					✓
	RELAYSENSOR				✓	
	IR_BLASTER				✓	
	IRTRIPORT				✓	
	IRTRIPORT_BLASTER				✓	
	SWITCH					✓
	subtype	port subtype				✓
	DIGITAL					SENSOR[3]
	IN					IR[3]
	OUT					IR[3]
	RS232					SERIAL[3]
	RS485					
	SPST_3A					RELAY[3]
	HDMI_3:1					SWITCH[3]

1. For Flex, only ***module*** 1 is enumerated. This is a known bug which currently affects only the Flex Link Relay & Sensor Cable (FLC-RS) where sensors are on module 2.
2. For Flex, ***ports*** is always 1. This is a known bug.
3. Specifies the ***type*** value(s) this ***subtype*** is associated with.

examples

GC-100-12

```
getdevices↵
device,1,1 SERIAL↵
device,2,1 SERIAL↵
device,3,3 RELAY↵
device,4,3 IR↵
device,5,3 IR↵
endlistdevices↵
```


Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

iTach IP2IR

```
getdevices↵
device,0,0 ETHERNET↵
device,1,3 IR↵
endlistdevices↵
```

Flex WiFi configured for Flex Link RS232 Serial Cable

```
getdevices↵
device,0,0 WIFI↵
device,1,1 SERIAL↵
endlistdevices↵
```

Global Connect Relay 6-port

```
getdevices↵
device,0,0 MODULE↵
device,1,6 RELAY_SPST_3A↵
endlistdevices↵
```

blink - blink device LED

The **blink** command is used to enable or disable a continuous blink sequence on the power LED of the device.

			Product Family			
			GC-100	iTach	Flex	Global Connect
command	**blink**,<*mode*>		✓			
response		(no response)	1			
parameters	***mode***	blink mode				
	0	blink disabled	✓*			
	1	blink enabled	✓			

1. GC-100 sends no response to this command.

examples

GC-100-12 - enable blink for power LED

```
blink,1↵
```

GC-100-12 - disable blink for power LED

```
blink,0↵
```


Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

get_NET - get network configuration
set_NET - set network configuration

The **get_NET** and **set_NET** commands allow managing the network configuration, and locking of device settings.

			Product Family			
			GC-100	iTach	Flex	Global Connect
command	**get_NET,*<module>*:*<port>***		✓	✓	✓	✓
	set_NET,*<module>*:*<port>*,*<cfglock>*, *<IPconfig>*,*<IPaddr>*,*<subnet>*,*<gateway>*		✓	[1]	[1]	[1]
response	**NET,*<module>*:*<port>*,*<cfglock>*, *<IPconfig>*,*<IPaddr>*,*<subnet>*,*<gateway>***		✓	✓	✓	✓
parameters	***module***	module address	0	0	0	0
	port	port/connector address	1	1	1	1
	cfglock	configuration lock setting				
	UNLOCKED		✓*	✓*	✓*	✓*
	LOCKED		✓	✓	✓	✓
	IPconfig	IP address assignment mode				
	STATIC		✓	✓	✓	✓
	DHCP		✓*	✓*	✓*	✓*
	IPaddr	IPV4 address	✓	✓	✓	✓
	subnet	IPV4 subnet	✓	✓	✓	✓
	gateway	IPV4 gateway	✓	✓	✓	✓

1. The **set_NET** command is not currently supported for iTach, Flex, or Global Connect.

examples

Global Connect

```
get_NET,0:1↵
NET,0:1,UNLOCKED,DHCP,192.168.0.100,255.255.255.0,192.168.0.1↵
```

GC-100-12

```
set_NET,0:1,UNLOCKED,STATIC,192.168.0.50,255.255.255.0,192.168.0.1↵
NET,0:1,UNLOCKED,STATIC,192.168.0.50,255.255.255.0,192.168.0.1↵
```


Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

4.4 CLASS-SPECIFIC COMMANDS

Global Caché products consist of one or more modules of different classes (types). Each of these module classes provides unique functionality and capabilities with associated unique API commands. The following subsections describe the currently available module classes and their available commands, including Infrared, Serial, Relay, Sensor, and Matrix/Switch classes.

4.4.1 INFRARED (IR)

The Global Caché Infrared (IR) module class includes the functions of IR Output (emitters and blasters), Sensor Inputs, and IR Input/Receive.

get_IR - get IR port mode
set_IR - set IR port mode

The **get_IR** and **set_IR** commands manage the functional mode of IR ports.

command
get_IR,*<module>*:*<port>*
set_IR,*<module>*:*<port>*,*<mode>*

response
IR,*<module>*:*<port>*,*<mode>*

parameters

			Product Family			
			GC-100	iTach	Flex	Global Connect
module	module address		2 | 4 | 5	1	1	1
port	port/connector address		1 | …| 3	1 | …| 3	1 [4]	1 | …| 3
mode	port I/O mode					
	IR	Infrared emitter	✓*	✓*[2]	✓	✓*
	BL2_BLASTER	Infrared blaster	✓			
	IR_NOCARRIER	Infrared envelope	✓			
	IR_BLASTER	Infrared blaster		✓*[2,3]	✓	✓[3]
	IRTRIPORT	Flex Link 3 Emitter or Tri-port Cable			✓	
	IRTRIPORT_BLASTER	Flex Link 2 Emitter 1 Blaster Cable			✓*	
	SENSOR	Sensor, polled	✓	✓	✓	✓
	SENSOR_NOTIFY	Sensor, change-notification	✓	✓	✓	✓
	SERIAL	RS232 or RS485 serial		✓	✓	✓
	RECEIVER	Infrared receiver				✓
	LED_LIGHTING			✓		

command: -06, -12, -18 [1] | IP2IR, WF2IR | [4] | IR

1. GC-100 resets after a successful **set_IR** request.
2. iTach default is **IR** for ports 1 and 2, and **IR_BLASTER** for port 3.
3. For iTach and Global Connect, IR_BLASTER is supported only on port 3.
4. Unique to the Flex product line are its various fixed-function Flex Link cables. The **set_IR** and **get_IR** commands set/get which of these Flex Link cables is connected, but do not control the mode of individual ports. As such, the ***port*** parameter has no effect (and must always be "1").

examples

Query setting for module 1, port 1

```
get_IR,1:1↵
IR,1:1,SENSOR↵
```


Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

GC-100-12 - set IR module 4 port 2 to Sensor Input mode (no change notification)

```
set_IR,4:2,SENSOR↵
IR,4:2,SENSOR↵
```

iTach - set port 3 to Blaster mode

```
set_IR,1:3,BLASTER↵
IR,1:3,BLASTER↵
```

Flex - set Flex Link cable mode to Serial

```
set_IR,1:1,SERIAL↵
IR,1:3,SERIAL↵
```

Global Connect - set port 1 to Sensor Input mode with change-notification

```
set_IR,1:1,SENSOR_NOTIFY↵
IR,1:1,SENSOR_NOTIFY↵
```

SENDING IR

Every IR device has IR codes associated with each of its functions. Each of the IR codes describes the timing and pattern of a specific pulse-modulated waveform. The IR code defines the electrical waveform, which is output from an IR port, emitted as an IR signal by an emitter or blaster, and finally received and interpreted by the IR device that triggers a function.

IR codes are typically represented and stored as text strings in various formats. IR codes for thousands of devices can be found in Control Tower, Global Caché's IR database in the cloud. However, in exceptional cases where a device's codes are not available in the database, they can usually be learned using the handheld IR remote and a Global Caché IR learner (see **get_IRL** command).

For additional details and discussion about IR signal structure, IR code formats, and IR parameters, see Appendix A.

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

sendir - transmit IR code

The **sendir** command is used to transmit infrared signals for controlling various functions of an IR device.

Processing of a **sendir** request starts only after the entire request is received, including parameters, code, and carriage-return. Then, if the requested port, format, and parameters are valid, the resulting action is determined by the current state of the requested port, as follows:

State of Requested Port	Action
idle	begin IR code transmission and respond with **completeir** when complete
busy transmitting same IR code	add or reset repeat value (see Smooth Continuous IR Repeat for more details)
busy transmitting different IR code	respond with **busyir**

			Product Family			
			GC-100	iTach	Flex	Global Connect
command	**sendir,<*module*>:<*port*>,<*ID*>,<*freq*>,<*repeat*>,<*offset*>,<*on1*>,<*off1*>,<*on2*>,<*off2*>,…,<*onN*>,<*offN*>**		-06, -12, -18	IP2IR, WF2IR	GC-IR-E, FLC-BL, FLC-3E, FLC-T3, FLC-2E1B	IR
response	**completeir,<*module*>:<*port*>,<*ID*>**					
	busyir					
parameters	***module***	module address	2 | 4 | 5	1	1	1
	port	port/connector address	1 | … | 3	1 | … | 3	1 | … | 3	1 | … | 3
	ID	code ID	0 | … | 65535	0 | … | 65535	0 | … | 65535	0 | … | 65535
	Freq	carrier frequency, Hz	20000 | … | 500000	15000 | … | 500000	15000 | … | 500000	15000 | … | 500000
	repeat	repeat count	1 | … | 31	1 | … | 50	1 | … | 20	1 | … | 20
	offset	preamble offset (if *repeat* > 1)	1 | … | 255	1 | … | 383	1 | … | 383	1 | … | 383
	onX	pulse count	1 | … | 50000	1 | … | 50000	4 | … | 50000	4 | … | 50000
	offX	gap count	1 | … | 50000	1 | … | 50000	4 | … | 50000	4 | … | 50000

examples

```
sendir,1:1,123,40000,1,1,96,24,48,24,24,24,48,24,24,24,48,24,24,24,24,24,48,24,24,24,24,24,24,24,1035↵
completeir,1:1,123↵
```

stopir - stop IR transmission

The **stopir** command is used to terminate an active IR transmission. This might be necessary, for example, if excessive repeats were initiated and are actively transmitting, but are not desired.

For all products except GC-100, the exact **stopir** request string is echoed back in a response to the originator of the **stopir** request.

If an IR transmission is active on the specified port it is immediately stopped, and the response is also sent to the originator of the stopped IR request (if different from the originator of the **stopir** request).

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

			Product Family			
			GC-100	iTach	Flex	Global Connect
command	**stopir,*<module>*:*<port>***		-06, -12, -18	IP2IR, WF2IR	GC-IR-E, FLC-BL, FLC-3E, FLC-T3, FLC-2E1B	IR
response	**stopir,*<module>*:*<port>***					
parameters	***module***	module address	2 | 4 | 5	1	1	1
	port	port/connector address	1 | … | 3	1 | … | 3	1 | … | 3	1 | … | 3

examples

GC-100

```
sendir,1:1,123,40000,10,1,96,24,48,24,24,24,48,24,24,24,48,24,24,24,24,24,48,24,24,24,24,24,24,24,24,1035↵
stopir,1:1↵
```

iTach, Flex, or Global Connect

```
sendir,1:1,123,40000,10,1,96,24,48,24,24,24,48,24,24,24,48,24,24,24,24,24,48,24,24,24,24,24,24,24,24,1035↵
stopir,1:1↵
stopir,1:1↵
```

RECEIVING IR

Some devices provide the ability to receive IR signals. This functionality is available in two forms, each with specific characteristics and purposes, as follows.

- IR Learner – Capable of receiving signals from an IR source within a very short distance (2.5 cm or less). It provides exceptional accuracy and sensitivity specifically for the purpose of learning IR codes from handheld remotes for later (re)transmission from an IR class module.
- IR Receiver – Capable of receiving signals from an IR source within a significant distance (up to 10 meters). It is not as accurate as the IR Learner, but can be used for IR gateway functionality, allowing handheld remotes or other IR sources to control other devices connected to the network. Note that this requires an application to process the received IR data and match against a list of known IR codes.

The API commands associated with these two forms of receiving IR are specified and described below.

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

get_IRL - enable IR Learner
stop_IRL - disable IR Learner

The IR Learner is enabled by sending a **get_IRL** request.

While the IR learner is enabled, each complete received IR sequence is converted into a formatted IR code string and sent to the client that originated the **get_IRL** command as an IR code string in Global Caché IR Format with module and port address parameters fixed at a value of 1.

The IR learner remains enabled until the originating client sends a **stop_IRL** request, or the client connection is closed.

		Product Family			
		GC-100	iTach	Flex	Global Connect
command	**get_IRL**				
response	**IR Learner Enabled** ***<IR_code>*** **…**		✓	✓	IR
command	**stop_IRL**		✓	✓	IR
response	**IR Learner Disabled**				
parameters	(see **sendir** command parameters)		✓	✓	IR

examples

```
get_IRL↵
IR Learner Enabled↵
```
(Aim handheld IR remote output to within 1” of Global Caché IR Learner, and press desired function button.)
```
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↵
IR Learner Disabled↵
```


Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

receiveIR - enable/disable IR input

The **receiveIR** command enables and disables IR receive mode on a supported port. Module and port address are required since a module may have multiple ports that support IR receive. Since some ports support multiple operation modes a port must first be configured for IR receive before IR receive mode can be enabled (see **set_IR** command).

While IR receive mode is enabled, any received IR signals are converted to a formatted IR code string and streamed to the client that originated the **receiveIR** request in Global Caché IR Format*.

* Currently, the port address parameter is always 1. This will be corrected in a future release to reflect the actual port address the IR code was received on.

IR receive mode remains enabled until the originating client disables IR receive mode or the connection is closed.

			Product Family			
			GC-100	iTach	Flex	Global Connect
command	**receiveIR,***<module>*:*<port>*,*<mode>*					
response	**receiveIR,***<module>*:*<port>*,*<mode>* *<IR_code>* …					IR
parameters	***module***	module address				1
	port	port address				1 | … | 3
	mode	IR receive mode				
	enabled	enable IR receive mode				✓
	disabled	disable IR receive mode				✓
	IR_code	IR code in Global Caché format				✓

examples

Configure port 2 for IR receive and enable IR receive mode

```
set_IR,1:2,RECEIVER↵
IR,1:2,RECEIVER↵
receiveIR,1:2,enabled↵
receiveIR,1:2,enabled↵
```

(Aim handheld remote at IR receiver and press a button.)

```
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↵
```

Disable IR receive mode on port 2

```
receiveIR,1:2,disabled↵
receiveIR,1:2,disabled↵
```


Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

4.4.2 SERIAL

The Global Caché Serial module class provides bidirectional transparent bridging between network and serial allowing any device with a serial UART interface to be controlled and monitored from the network.

To utilize this functionality, the Serial module class is unique in that it provides a dedicated TCP socket server for each available serial port. The socket server supports multiple simultaneous raw TCP connections, allowing multiple clients to achieve bidirectional communication with the associated serial port. Data transfer is transparent (not interpreted or altered in any way).

TCP port numbers and maximum simultaneous connections per TCP port are shown below for each product line.

	GC-100-06	GC-100-12 GC-100-18	iTach	Flex	Global Connect
TCP port	4999	4999, 5000	4999	4999	4999
Maximum Connections	1	1	4*	8	8

* iTach multiport mode must be enabled in the configuration webpage **Serial** module settings.

To establish bidirectional direct communication with a device connected to the serial port, a network client simply opens a raw TCP connection to the specified server port.

In applications where multiple clients may simultaneously communicate with the serial device, several important details and constraints should be considered in order to achieve efficient and successful communications. Please see Appendix B for an in-depth discussion of this topic.

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

get_SERIAL - get Serial port configuration
set_SERIAL - set Serial port configuration

The **get_SERIAL** and **set_SERIAL** commands are used to manage the configuration of a serial port.

			Product Family			
			GC-100	iTach	Flex	Global Connect
command	**get_SERIAL,*<module>*:*<port>***					
	set_SERIAL,*<module>*:*<port>*, *<baudrate>*,*<flowcontrol/duplex>*,*<parity>*,*[stopbits]*		-06, -12, -18	IP2SL WF2SL	FLC-SL-232 FLC-SL-MJ FLC-SL-485	SL
response	**SERIAL,*<module>*:*<port>*, *<baudrate>*,*<flowcontrol/duplex>*,*<parity>*,*[stopbits]***					
parameters	***module***	module address	1 | 2	1	1	1
	port	port/connector address	1	1	1	1
	baudrate	baud rate, bits per second	[1]	[1]	300 | … | 115200[2]	300 | … | 115200[2]
	1200		✓	✓		
	2400		✓	✓		
	4800		✓	✓		
	9600		✓	✓		
	14400			✓		
	19200		✓*	✓*	*	*
	38400		✓	✓		
	57600		✓	✓		
	115200			✓		
	flowcontrol	flow control mode			FLC-SL-232 FLC-SL-MJ	
	FLOW_NONE		✓*	✓*	✓*	✓*
	FLOW_HARDWARE		✓	✓	✓	✓
	duplex	duplex mode			FLC-SL-485	
	DUPLEX_HALF				✓	
	DUPLEX_FULL				✓*	
	parity	parity bit mode				
	PARITY_NO		✓*	✓*	✓*	✓*
	PARITY_ODD		✓	✓	✓	✓
	PARITY_EVEN		✓	✓	✓	✓
	stopbits	stopbit mode				
	STOPBITS_1		✓*	✓*	✓*	✓*
	STOPBITS_2		✓	✓	✓	✓

1. Only the indicated specific baud rate values are supported.
2. All baud rates in the specified range are supported.

examples

GC-100-06 or iTach IP2SL

```
get_SERIAL,1:1↵
SERIAL,1:1,19200,FLOW_NONE,PARITY_NO↵

set_SERIAL,1:1,38400,FLOW_HARDWARE,PARITY_EVEN↵
SERIAL,1:1,38400,FLOW_HARDWARE,PARITY_EVEN↵
```

Flex with Flex Link RS232 cable, or Global Connect Serial

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

set_SERIAL,1:1,9600,FLOW_HARDWARE,PARITY_EVEN,STOPBITS_2↵
SERIAL,1:1,9600,FLOW_HARDWARE,PARITY_EVEN,STOPBITS_2↵
```


Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

4.4.3 RELAYS

The Global Caché Relay module class provides relay outputs that allow network-controlled dry contact-closure.

get_RELAY - get configurable relay type
set_RELAY - set configurable relay type

The **get_RELAY** and **set_RELAY** commands apply to Relay class modules having configurable relays and are used to read and modify relay configuration. In the context of the API, a configurable relay is represented as logical relays which is an abstraction of physical relay ports. To best understand this logical relay abstraction and the associated relay configurations and types, please refer to the details in Appendix D.

Note: Configurable relays are currently available only on the Flex products when used with the Flex Link Relay & Sensor Cable.

			Product Family			
			GC-100	iTach	Flex	Global Connect
command	**get_RELAY,*<module>*:*<address>***					
	set_RELAY,*<module>*:*<address>*,*<type>*				FLC-RS	
response	**RELAY,*<module>*:*<port>*,*<type>***					
parameters	***module***	module address			1	
	address	logical relay address			1 | … | 4	
	type [1]	relay type				
	SPST	Single Position, Single Throw			✓	
	SPDT	Single Position Double Throw			✓	
	DPDT	Double Position, Double Throw			✓	
	Disabled	Relay port disabled			✓*	
	Unavailable [2]	Relay port in use			✓	

1. Valid ***<type>*** values depend on the active relay configuration. See Appendix D for detailed explanation on the required process for (re)configuring relays.
2. This is a response value only and cannot be used in the request.

examples

Flex with Flex Link Relay & Sensor Cable - configure logical relay 1 as SPST

```
get_RELAY,1:1↵   First, check port 1 to ensure it is Disabled)
RELAY,1:1,Disabled↵
set_RELAY,1:1,SPST↵
RELAY,1:1,SPST↵
```

Flex with Flex Link Relay & Sensor Cable - configure logical relay type 1 as SPDT

```
set_RELAY,1:1,Disabled↵   (Set ports 1 and 2 to Disabled before reconfiguring to SPDT.)
RELAY,1:1,Disabled↵
set_RELAY,1:2,Disabled↵
RELAY,1:2,Disabled↵
set_RELAY,1:1,SPDT↵
RELAY,1:1,SPDT↵
```


Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

getstate - get state of relay output
setstate - set state of relay output

The **getstate** and **setstate** commands are used to poll, monitor, and control the state of Relay module outputs.

The **getstate** command's optional ***mode*** parameter supports the **notify** option which enables change notification upon any state change for the specified port. See Appendix C for more details.

			Product Family			
			GC-100	iTach	Flex	Global Connect
command	**getstate,*<module>*:*<port>*[,*mode*]**					
	setstate,*<module>*:*<port>*,*<state>*		-12, -18	WF2CC, IP2CC	FLC-RS	RT
response	**state,*<module>*:*<port>*,*<state>***					
parameters	***module***	module address	3	1	1	1
	port	port/connector address	1 | … | 3	1 | … | 3	1 | … | 4	1 | … | 6
	state[1]	relay state				
	0	off (open)	✓*	✓*	✓*	✓*
	1	SPST - on (closed) SPDT/DPDT - on1 (1st throw/position)	✓[2]	✓[2]	✓[2,3]	✓[2]
	2	SPDT/DPDT - on2 (2nd throw/position)			✓	
	mode	optional get mode setting				
	notify	enable state change notification				✓

1. The ***state*** value (state of the relay) is not persistent through a device reset or power-cycle, and thus always reverts to the default value (0/off/open) after a reset or power-cycle.
2. SPST - on (closed)
3. SPDT/DPDT - on1 (1st throw/position)

examples

GC-100-12 - set relay port 2 = off (open):

```
setstate,3:2,0↵
state,3:2,0↵
```

Flex - set logical relay port 3 (SPST) = on (closed):

```
setstate,1:3,1↵
state,1:3,1↵
```

Flex - set logical relay port 1 (SPDT) = on2 (2nd throw/position):

```
setstate,1:1,2↵
state,1:1,2↵
```


Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

4.4.4 SENSORS

The Global Caché Sensors module class provides bistate (digital) sensor inputs. The currently available products allow detection of contact-closure and presence of voltage, current, or video.

getstate - get state of sensor input

The **getstate** command is used to poll and monitor the state of Sensor module input ports.

The **getstate** command's optional **mode** parameter supports the **notify** option which enables change notification for any state change for the specified port. See Appendix C for more details.

			Product Family			
			GC-100	iTach	Flex	Global Connect
command	**getstate,*<module>*:*<port>*[,*mode*]**		-06, -12, -18	WF2IR, IP2IR	FLC-RS	IR
response	**state,*<module>*:*<port>*,*<state>***					
parameters	***module***	module address	3 | 4	1	2	1
	port	port/connector address	1 | … | 3	1 | … | 3	1 | … | 4	1 | … | 3
	state	sensor state				
	0	off (open)	✓	✓	✓	✓
	1	on (closed)	✓	✓	✓	✓
	mode	optional get mode setting				
	notify	enable state change notification	[1]			✓

1. GC-100 state change notification over TCP is supported via the **statechange** response message. See Appendix C for additional details

examples

GC-100-06 - get module 3 sensor port 2 state

```
getstate,3:2↵
state,3:2,1↵
```

Flex - get sensor port 3 state

```
getstate,2:3↵
state,2:3,0↵
```


Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

SENSOR CHANGE NOTIFICATION

get_SENSORNOTIFY - get sensor-notify settings
set_SENSORNOTIFY - set sensor-notify settings

The **set_SENSORNOTIFY** command is available on some Sensor class modules. It allows control and configuration of the UDP and TCP Change Notification features for sensor-inputs, if supported. For a detailed explanation of Change Notification, see Appendix C.

			Product Family			
			GC-100	iTach	Flex	Global Connect
command	**get_SENSORNOTIFY,*<module>*:*<port>***				✓	✓
	set_SENSORNOTIFY,*<module>*:*<port>*,*<n_port>*,*<n_interval>*,*[debounce]*					
response	**SENSORNOTIFY,*<module>*:*<port>*,*<n_port>*,*<n_interval>*,*[debounce]***				✓	✓
parameters	***module***	module address			2	1
	port	port/connector address			1 | … | 4	1 | … | 3
	n_port	notify port				
	0	all notifications disabled			✓	✓
	1 | … | 65535	UDP port number			✓	✓
	n_interval	notify interval				
	0	periodic notifications disabled			✓	✓
	1 | … | 65535	time, seconds			✓	✓
	debounce	minimum duration of valid state				10us | … | 1s[1]

1. The ***debounce*** field allows specifying time units in standard abbreviated notations including "us", "ms", and "s" (microseconds, milliseconds, and seconds, respectively). If no units are specified, "s" is assumed. If no value is specified, a default value of 100 ms is used.

examples

Flex - disable all notification for sensor port 1:

```
set_SENSORNOTIFY,1:2,0,0↵
SENSORNOTIFY,1:2,0,0↵
```

Flex - enable change notification for sensor port 2, at UDP port 12345:

```
set_SENSORNOTIFY,1:2,12345,0↵
SENSORNOTIFY,1:2,12345,0↵
```

Global Connect - enable change notification ***and*** periodic notification for sensor port 1, at UDP port 54321

```
set_SENSORNOTIFY,1:1,54321,10↵
SENSORNOTIFY,1:1,12345,10,100ms↵
```

Global Connect - enable change-notification for sensor port 1, at UDP port 12345, with debounce value of 500us.

```
set_SENSORNOTIFY,1:1,12345,10,500us↵
SENSORNOTIFY,1:1,12345,10,500us↵
```


Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

4.4.5 MATRIX/SWITCH

The Global Caché Matrix/Switch module class provides matrix signal switching. Current available products include a 3 input, 1 output HDMI switch with bidirectional CEC communication and active port detection.

getstate - get state of matrix switcher input-output selection
setstate - set state of matrix switcher input-output selection

The **getstate** and **setstate** commands are used to poll, monitor, and set the input/output selection state of the matrix switcher.

Notes:

- Selecting an input/output disables the previous selected input/output.
- Disabling a selected input also disables the selected output (currently this occurs even if the input is already disabled, but this is unintended and will be corrected in a future update).
- Disabling a selected output also disables the selected input.

The **getstate** command's optional ***mode*** parameter supports the **notify** option which enables change notification for any state change for the specified port. See Appendix C for more details.

			Product Family			
			GC-100	iTach	Flex	Global Connect
command	**getstate,*<module>*:*<in_port>*[,*mode*]**					
	setstate,*<module>*:*<in_port>*,*<out_port>*					HM
response	**state,*<module>*:*<in_port>*,*<state>***					
parameters	***module***	module address				1
	in_port	input port address				
	1 | … | N	input port address, where N = number of input ports				1 | … | 3
	0	disable the specified ***<out_port>***				0*
	out_port	output port address				
	1 | … | N	output port address, where N = number of output ports				1
	0	disable the specified ***<in_port>***				0*
	mode	optional get mode setting				
	notify	enable state change notification				✓

examples

Set input port 1 linked to output port 1
```
setstate,1:1,1↵
state,1:1,1↵
```

Set input port 2 linked to output port 1
```
setstate,1:2,1↵
state,1:2,1↵
```

Get input port 3 state
```
getstate,1:3↵
state,1:3,0↵    (input port 3 = disabled)
```


Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

getactive - get active state of matrix switcher ports

The **getactive** command is used to poll the active state of matrix switcher ports.

A port's active state is determined by whether an active (powered on) HDMI device is connected to the port.

			Product Family			
			GC-100	iTach	Flex	Global Connect
command	**getactive,*<module>***					
response	**active,*<module>*** **OUT:** ***<out_port1>*,*<state>*** … ***<out_portN>*,*<state>*** **IN:** ***<in_port1>*,*<state>*** … ***<in_portN>*,*<state>*** **endactive,*<module>***					HM
parameters	***module***	module address				1
	state	port state				
	true	active				✓
	false	inactive				✓
	in_port	input port address				
	1 | … | N	input port address, where N = number of input ports				1 | … | 3
	out_port	output port address				
	1 | … | N	output port address, where N = number of output ports				1

examples

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


Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

HDMI CEC

CEC - send/receive HDMI CEC messages

The **CEC** command allows for network control, query, and monitoring of devices interconnected via the HDMI switch.

Note: Only devices connected to active/selected switch ports are accessible via CEC. In other words, a device connected to an unselected switch input or output port is not connected to the CEC bus and cannot be accessed via CEC. For more information on controlling switch port selection, see the Matrix/Switch class **`getstate`** and **`setstate`** commands.

The **CEC** command provides two modes of operation:

- TX - transmit mode allows control and query of HDMI devices.
- RX - receive mode allows monitoring messages sent and received by HDMI devices (including from the switch itself).

The behavior of the ***<acknowledge>*** parameter in the **CEC** command's response depends on the type of CEC message:

- Unicast messages must be acknowledged (ACK'd) by the destination device. If successfully acknowledged, **ACK** is returned as the ***<acknowledge>*** response parameter value. Up to two (2) retries are attempted. If not acknowledged after three (3) total transmit attempts, "**,!ACK**" is returned as the ***<acknowledge>*** response parameter value, followed by a CEC transmit error message (see API Errors).

- Broadcast messages (sent to logical address "F") need not be ACK'd by receiving device(s) but may be NACK'd (rejected) by receiving device(s). Any such NACK'd message will yield **NACK** returned as the ***<acknowledge>*** response parameter value. If no NACK occurs, the broadcast message is considered accepted and successful, and no value is returned for the ***<acknowledge>*** response parameter value.

Format specification and examples for the **TX** and **RX** operation modes are shown separately below.

			Product Family			
			GC-100	iTach	Flex	Global Connect
command	**CEC,*<module>*:*<out_port>*,TX,*<message>***					
						SW
response	**CEC,*<module>*:*<out_port>*,TX,*<message>*,*<acknowledge>***					
parameters	***module***	module address				1
	out_port	output port address				1
	message	CEC message, colon-delimited ASCII hex bytes[1]				
	<byte1>: ... :<byteN>	byteX = ASCII hex, N <= 16				✓
	acknowledge					
	ACK	unicast message acknowledged				✓
	!ACK	unicast message not acknowledged				✓
	NACK	broadcast message rejected				✓
	(none)[2]	broadcast message accepted				✓

1. See Colon-Delimited Message Format for a detailed format description.
2. No acknowledge value is shown for an accepted (not explicitly rejected) broadcast message. However, currently an unintended trailing comma is output. This will be corrected in a future firmware update.

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

examples

Playback 1 to TV – Image View On ... turns display on

```
CEC,1:1,TX,40:04↵
CEC,1:1,TX,40:04,ACK↵
```

Playback 1 to TV – Image View On ... turns display on

```
CEC,1:1,TX,40:04↵
CEC,1:1,TX,40:04,ACK↵
```

Failed (not ACK'd) unicast message

```
CEC,1:1,TX,40:04↵
CEC,1:1,TX,40:04,!ACK↵
CEC,1:1,TX,40:04,!ACK↵
CEC,1:1,TX,40:04,!ACK↵
ERR SW011↵
```

command **CEC,*<module>*:*<out_port>*,RX,*<enable>***

response **CEC,*<module>*:*<out_port>*,RX,*<message>*,*<acknowledge>***

parameters			Product Family: GC-100	iTach	Flex	Global Connect
						HM
	module	module address				1
	out_port	output port address				1
	enable	receive mode				
	enabled on 1	enable monitoring of CEC messages				✓
	disabled off 0	disable monitoring of CEC messages				✓
	message	CEC message, colon-delimited ASCII hex bytes[1]				
	<byte1>: ... :<byteN>	byteX = ASCII hex byte, N <= 16				✓
	acknowledge					
	ACK	unicast message acknowledged				✓
	!ACK	unicast message not acknowledged				✓
	NACK	broadcast message rejected				✓
	(none)[2]	broadcast message accepted				✓

1. See Colon-Delimited Message Format for a detailed format description.
2. No acknowledge value is returned for an accepted (not explicitly rejected) broadcast message. However, currently an unintended trailing comma is output. This will be fixed in a future firmware update.

examples

Enable RX mode

```
CEC,1:1,RX,1↵
CEC,1:1,RX,enabled↵
```

Observe successful (ACK'd) message

```
CEC,1:1,RX,10:20:30:40,ACK↵
```

Observe failed (not ACK'd) message

```
CEC,1:1,RX,10:20:30:40,!ACK↵
CEC,1:1,RX,10:20:30:40,!ACK↵
```


Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

```
CEC,1:1,RX,10:20:30:40,!ACK↵
```

Observe successful (not NACK'd) broadcast message

```
CEC,1:1,RX,1F:20:30:40↵
```

Observe failed (NACK'd) broadcast message

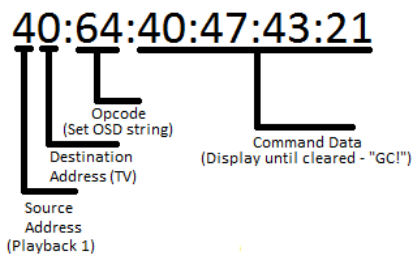
```
CEC,1:1,RX,10:20:30:40,!ACK↵
CEC,1:1,RX,10:20:30:40,!ACK↵
```

CEC COLON-DELIMITED MESSAGE FORMAT

The ***<message>*** parameter of the CEC request/response is formatted as a sequence of colon-delimited hexadecimal byte values which represent the actual transmitted or received CEC message.

The first byte of the message indicates the logical address of the source and destination devices. The second byte indicates the opcode/command. The remaining bytes (if necessitated by the opcode) are the message data.

Below is an example CEC command, sent from the Playback 1 device to the TV, to display the string "GC!" on the TV:



(A complete list of CEC commands can be found in the latest CEC specification document.)

The CEC logical addresses are listed below. These addresses are for use in the source and destination address in the CEC message. Logical addresses are represented by hexadecimal characters. CEC devices 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
9 (0x9)	Recording Device 3
10 (0xA)	Tuner 4

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

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)

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

APPENDIX A – INFRARED SIGNALING AND CODES

IR SIGNALING

IR transmissions are created by sending a pattern of timed pulses modulated on a specific carrier frequency (ƒ). This pattern of pulses is comprised of a series of ***<on>*** and ***<off>*** states, where the carrier frequency (ƒ) is present during the ***<on>*** state, and absent during the ***<off>*** state (also sometimes called the gap state). The carrier frequency is typically between 35 to 45 kHz, but in exceptional cases can reach 500kHz.

The duration of each ***<on>*** and ***<off>*** state is represented as a pair of pulse count values, sometimes called a pulse pair. A pulse count value is the count of carrier frequency periods that elapses during the ***<on>*** or ***<off>*** state. For example, an ***<off>*** value (pulse count) of 24 modulated on a carrier frequency of 40 kHz results in a duration of 600µS, as calculated below.

`period = 1/ƒ = 1/40 kHz = 1/40000 = 0.000025 seconds = 25µs`

<off> `value = 24 pulses, so` ***<off>*** `duration = 24 × 25µs = 600µs`

Figure A-1 shows an example IR sequence of "**4,5,6,5**" with corresponding waveform illustrating the ***<on>*** and ***<off>*** patterns.

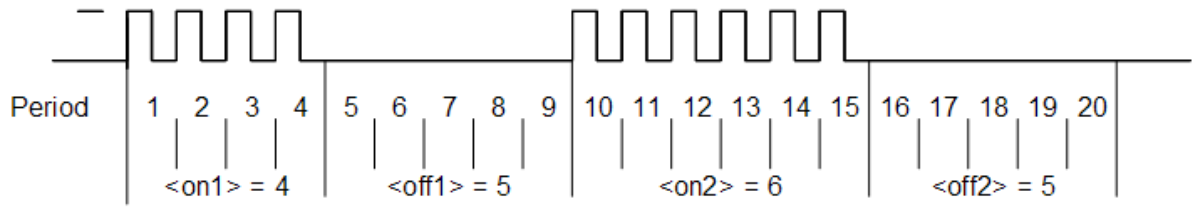


Figure A-1

IR sequences typically end with a relatively large <off> value. This serves as a resting or gap state which allows the receiving device to detect the end of the IR sequence and thereby distinguish it from any subsequent IR signal it may receive.

GLOBAL CACHÉ IR FORMAT

Global Caché's IR code format and parameters are described in detail below.

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

<module> - module address
The module parameter specifies the address of the module which contains the target IR output port.

<port> - port address
The port parameter specifies the address of the target IR output port.

<ID> - ID number
The ID parameter is an arbitrary number assigned by the requester. It is included in the **completeir** response when the IR transmit is successfully completed. Applications can use this to identify when a particular IR code has completed.

<freq> - IR carrier frequency, Hz
The frequency parameter specifies the carrier frequency of the IR waveform. See the IR Signaling section for additional details.

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

<repeat> - repeat count
The repeat parameter defines the number of times to transmit the IR code. Although this value is set by the initial request, it can be affected during IR transmission in the following two ways.

- A **stopir** command unconditionally terminates an active IR transmission.
- An identical IR request received while the same IR code is already transmitting results in the original request's repeats being increased by the specified repeat value, or reset to the specified repeat value (see Smooth Continuous IR Repeat).

<offset> - preamble offset
The offset parameter is used only if the repeat value is greater than 1. If the IR code includes a preamble, the offset value defines the preamble by indicating the offset into the pulse-count sequence where a repeat starts. The offset value must always be an odd value, since an IR sequence must begin at the start of a pulse-pair. The following table shows the relationship between offset value and repeat start location (when repeat value is > 1).

<offset> (always odd)	***<repeat>*** starts at:	
1	***<on1>***	(no preamble)
3	***<on2>***	
…	…	
N–1	***<on((N/2)-1)>***	N = total number of pulse-pairs

<on1> - first on pulse count
The on parameter specifies the number of output pulses, that is the number of periods of the carrier frequency. See the IR Signaling section for additional details.

<off1> - first offpulse count
The off parameter specifies the duration of the gap (absence of pulses) by the number of periods of the carrier frequency. See the IR Signaling section for additional details.

<onN> and ***<offN>*** - the final pulse count values, where ***N*** = the number of pulse pairs.
Note: There must be an equal number of ***<on>*** and ***<off>*** states. Also, every ***<on>*** and ***<off>*** state must meet an 80µS minimum duration requirement.

Example: With a carrier frequency of 60 kHz, the minimum value for <on> and <off> states can be calculated, as shown below.

$$\textbf{<off>}_{min} = \textbf{<on>}_{min} \geqslant 80\mu S \times f = 80\mu S \times 60\ KHz = 4.8\ pulses$$

For accurate reproduction of an IR code at 60 kHz, all ***<on>*** and ***<off>*** pulse-counts in the timing pattern must be 5 or higher.

All the above conditions above must be satisfied for the IR code to be valid. If a variable is missing or out of range, an error will be returned.

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

GLOBAL CACHÉ COMPRESSED IR FORMAT

The compressed IR format is based on a technique of replacing up to 15 unique ***<on>*** and ***<off>*** pulse-pairs with a single uppercase alphanumeric character (e.g., A, B, C,…). The first time a unique pulse-pair occurs in the IR command, it is assigned a single alphanumeric character. Any subsequent occurrences of that same pulse-pair are then replaced with that assigned character. This allows the **sendir** request to be significantly compressed in length, which is advantageous in some cases, especially for very long IR codes.

Example IR code in standard Global Caché IR format:

```
sendir,1:1,123,40000,1,1,4,5,4,5,8,9,4,5,8,9,8,9↵
```

To convert to Global Caché Compressed format, we first find each unique pulse-pair (highlighted in blue).

```
sendir,1:1,123,40000,1,1,4,5,4,5,8,9,4,5,8,9,8,9↵
```

We then assign a single uppercase alphanumeric character to each unique pulse-pair. In this example, **A** is assigned to **4,5,** and **B** is assigned to **8,9,**. We then replace all subsequent occurrences of each unique pulse-pair (including commas) with its associated assigned alphanumeric character. This yields the resulting Global Caché Compressed format code as follows:

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

Note: The above example IR code yields a minor reduction in length when converted to Global Caché Compressed format. However, typical IR codes are much longer, and thus the reduction in code length is more significant.

SMOOTH CONTINUOUS IR REPEAT

Smooth Continuous IR Repeat is a feature which provides smooth sustained control functions (for example, volume control, channel up/down) without hesitations or intermittent response at the controlled device. This feature is supported on iTach and Flex. Support will be added for Global Connect in a future release (currently, Global Connect adds repeat values).

One approach to executing repeated IR functions is the scenario where a control application sends a **sendir** request with very large repeat count, then later terminates the request with a **stopir** request. However, this can result in uncontrolled behavior. For example, consider the user pressing and holding the Volume Up button. The control application sends a request with a very large repeat count, then waits for the user to release the button (at which time a **stopir** request will be sent to stop the original request). Suppose the control application's network connection is temporarily interrupted or unexpectedly closed while the user is holding the button. The IR code may continue repeating before the control application is able to reconnect and send a **stopir** command, resulting in a runaway scenario where the volume increases uncontrollably, possibly causing equipment damage.

To avoid the runaway scenario, the Global Caché device actively manages and limits the repeat count. This simply requires that the client choose an optimally minimal repeat value, and periodically (re)send the same IR request as long as the repeating function is asserted by the user (e.g., user is holding a button on a remote). Each time the repeated IR request is received by the Global Caché device, it resets the repeat count to the original requested value and continues transmitting the IR code. In this way, if the client connection is unexpectedly interrupted or disconnected, the IR code transmission stops after completing a minimal number of repeats, thereby mitigating the runaway scenario.

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

Example: The user presses and holds the Volume Up button. The control application responds by sending an IR request with repeat = 5. After a short delay, the control application detects the user is still holding the Volume Up button and resends the exact same IR request. At that moment, the IR code has completed three (3) repeats of the IR transmission, with two (2) remaining. But since the same IR request was received again, the repeat count is reset back to five (5), and the IR code transmission continues. This process continues indefinitely while the user holds the button, providing a smooth response at the controlled device. However, the command will repeat no more than five (5) times after the button is released, or after the client connection is unexpectedly interrupted or disconnected.

If each repeat IR request is received before the current request's repeats have completed transmitting, smooth operation is achieved at the controlled device. Ideally, the requested function (for example, Volume Up) should stop executing as soon as possible after the user stops pressing the button. This optimal (smallest) repeat value can be determined in part by considering the time required for request to be sent by the application/client on the network, any potential network latencies, and the time required to complete one transmission of the IR code. If the repeat value is too small, the IR request may complete before the next repeat request is sent and received, which introduces a small delay before the next repeat command is sent, which may cause occasional or continuous hesitations (observed at the controlled device) while the user holds the button.

APPENDIX B – SERIAL TO NETWORK BRIDGING

The Serial module class allows multiple network clients to simultaneously send and receive data to and from a single serial device. This is a complex process which must be carefully managed to maintain integrity of the communications between each network client and the serial device. The following sections describe this process for both directions of data transfer between the network and the serial device.

SENDING DATA TO A SERIAL DEVICE

The Serial module class achieves the functionality of multiple clients transmitting to a single serial device by carefully managing incoming network data according to the order and timing of received network packets, then transmitting that same data to the serial device in specifically managed and timed serial packets.

Consider a case where two TCP clients are connected to a Serial module, and each simultaneously send a command string for transmission to the connected serial device. **TCP client A** sends its command/data in two (2) separate network packets, while **TCP client B** sends its command/data in a single network packet. In this case, since **TCP client A** sent two (2) separate packets, it is entirely possible that the command/data from **TCP client B** is received and transmitted to the serial device in between the two (2) network packets sent by **TCP client A**. The resulting data transmitted to the serial device would then be an undefined combination of both commands/data, and very likely invalid.

This example illustrates why it is important for TCP clients to send complete command/data within a single network packet whenever possible. This ensures the command/data is received as complete and intact by the connected serial device.

RECEIVING DATA FROM A SERIAL DEVICE

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

The serial module achieves transmission of incoming serial data to multiple network clients by packetizing the incoming serial data according to timing and size.

Starting from an idle state, the first character received to the serial port marks the beginning of the incoming serial packet. Additional incoming serial data is added to the packet until one of the following conditions occurs:

a.) The serial receive buffer reaches near-full capacity.

b.) A time threshold (based on current baud rate) elapses with no characters being received.

If either of the above conditions are satisfied, the packet is considered complete, and it is immediately transmitted to all connected network clients.

APPENDIX C – CHANGE NOTIFICATION

The Change Notification feature is available on various module class I/O ports. It provides real-time streaming notification of port state changes.

A change notification message is simple and concise, indicating the module, port, and new state, as follows:

<state_response>,<module>:<port>,<state>

Delivery of change notification messages is achieved through several mechanisms based on different network protocols (depending on product line), including UDP multicast and TCP. These are discussed in the following subsections.

UDP

UDP change notification utilizes the Internet Group Management Protocol (IGMP) to send a multicast UDP message when an I/O port changes state, and/or at a configured time interval. This feature is supported on all products for Sensor class modules only. The format of the notification message follows exactly the Sensor class **getstate** command response.

Several configurable options are available which allow customization to accommodate various applications and environments. The following options are individually configurable for each port.

- Notify Port – destination port number for change notification messages.
- Notify Interval – time interval for periodic update messages, in seconds.

When selecting UDP port values, it is advisable to avoid conflicts with ports already in use in 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) 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 locally by other network devices.

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

TCP

GC-100 and Global Connect products support TCP change notification which provides real-time streaming notifications of I/O port state changes, delivered through a standard TCP API connection.

GC-100 change notification is supported only for Sensor class modules and must be enabled through the web configuration for the associated IR and sensors module. Selecting the Sensor with Auto-notify option for a port enables change notification mode for that port and any state change on that port results in a message being sent to any connected client(s) in the format shown below. (See the Sensor class **getstate** command for parameter descriptions.)

statechange,*<module>*:*<port>*,*<state>*

Global Connect change notification messages are enabled and delivered within the context of each client's TCP connection. Any client can request change notifications for any available I/O port (in a **getstate** request, with the optional ***mode*** parameter and the **notify** option). Multiple requests for any port can be active on one or more TCP connections. An enabled notification persists for the life of the TCP connection of the client that enabled the notification. Thus, any termination of a TCP connection terminates all active change notifications enabled for that connection but does not affect change notifications enabled in other client connections. The format of the notification message follows exactly the **getstate** command response for the associated module class.

DEBOUNCE

Some module classes support a debounce feature for I/O port state changes.

The debounce setting can be configured for each port. It establishes the minimum state duration required to trigger a change notification for that port. Thus, if a port state change occurs at a frequency which results in the duration of the port's state being less than the debounce value then the state change is ignored, and no change notification occurs. Conversely, if a port state change occurs and the new state persists for a duration greater than the debounce value, a change notification does occur.

APPENDIX D – CONFIGURABLE RELAYS

Some Relay class modules provide configurable relay functionality which supports various combinations of SPST, SPDT, or DPDT relay types. This configurability is accomplished via hardware jumpers and network API commands. Hardware jumper specifications for supported relay types can be found in the User Guide or Tech Guide for the configurable relay product. Software configuration, control, and monitoring of relay outputs is achieved via the TCP API commands specified in Section 4.4.3.

The purpose of the following subsections is to explain the concepts of **physical relay ports** and **logical relays** pertaining to configurable relays.

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

PHYSICAL PORTS VS. LOGICAL RELAYS

To understand configurable relays, a clarification must be made regarding the concepts of physical relay ports and logical relays.

Physical relay ports are the actual hardware relay outputs which are implemented by electromechanical relays. It is best to think of physical relay ports as not directly accessible or controllable via API commands, but instead as the underlying foundation for the abstracted logical relays which are configured, accessed, and controlled via API commands. Figure D-1 shows a visual representation of physical relay ports for an example Relay class module having four (4) relay outputs.

Figure D-1

Logical relays are an abstraction of the physical relay ports. Logical relays provide configurable capability to function as various relay types (SPST, SPDT, or DPDT). These relay types utilize one or more physical relay ports to achieve their functionality.

Key points to consider about logical relay abstraction are as follows:

1. Number of physical relay ports utilized by various logical relay types (referred to as the footprint).
2. Valid mappings (positions) for various logical relay types.
3. Address of valid logical relay mappings (for use in API commands).

The above points are visualized in the following **Figure D-2** and **Figure D-3**.

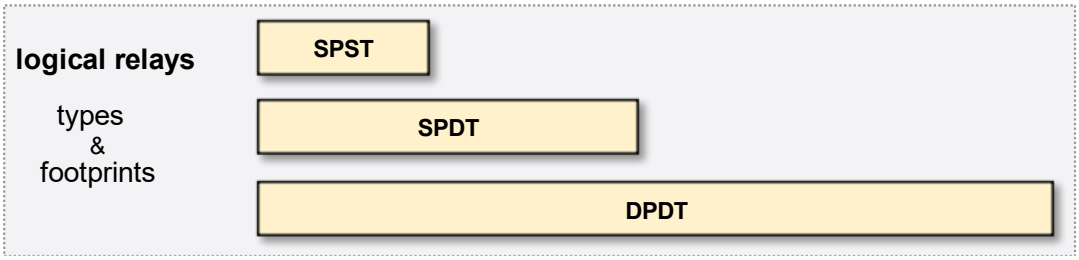


Figure D-2

Figure D-2 illustrates the number of physical relay ports occupied by various logical relay types (often referred to as the footprint)

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

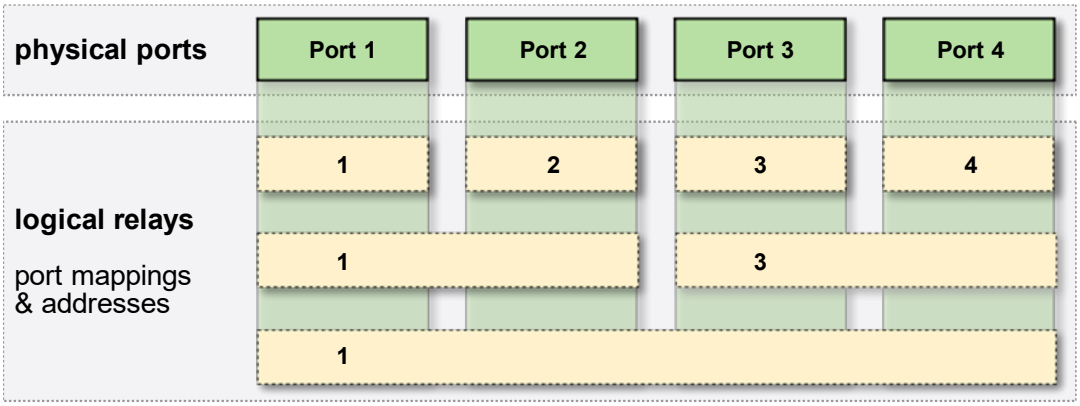


Figure D-3

Figure D-3 displays valid positions and addresses for various logical relay footprints juxtaposed against the underlying physical relay ports. Notice that logical relays always occupy one, two, or four (1, 2, or 4) physical relay ports. Also notice that logical relays occupying more than one (1) physical relay port are always combined in multiples of two (2), and are always aligned at an odd address.

LOGICAL RELAY TYPES

The relay types supported by Relay class modules with configurable relays are specified in the following table with description and footprint (number of physical relay ports utilized).

Relay type	Description	Footprint
SPST	Single Position Single Throw	1
SPDT	Single Position Double Throw	2
DPDT	Double Position Double Throw	4

CONFIGURING LOGICAL RELAYS

Configuration of logical relays is accomplished with the **get_RELAY** and **set_RELAY** commands. Specifications for these commands and associated parameters can be found in Section 4.4.3.

Figure D-4 illustrates several examples of valid logical relay configurations for a Relay class module with 4 physical ports. Note this is not an exhaustive list of all possible configurations.

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

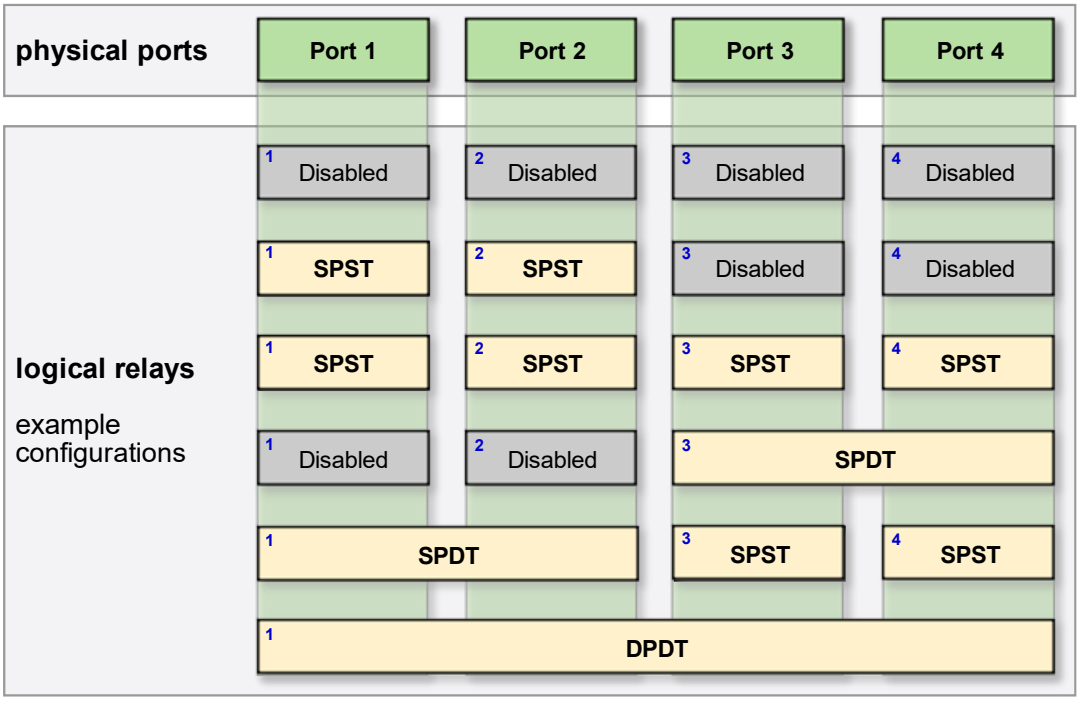


Figure D-4

In **Figure D-4**, logical relays are shown as yellow or gray rectangles containing the name of the relay type. The small blue number in the top-left corner indicates the logical relay's address, which is always the lowest port number of the ports occupied by the logical relay. Respectively, these two values are the ***<type>*** and ***<address>*** parameters to be used with the API commands **get_RELAY** and **set_RELAY** for configuration of the logical relays.

Also notice in **Figure D-4** some logical relays have a **Disabled** type. This represents the unconfigured state. It is the default setting at power-on or reset. More importantly, it is a prerequisite setting when changing the configuration of logical relays, described as follows.

When (re)configuring logical relays, the following order of steps must be taken:

1. Set all ports utilized by the desired logical relay to type **Disabled**.
2. Configure hardware jumpers for the desired relay type (see the User or Tech Guide for the product being used).
3. Connect external wiring from the relay terminals to the devices being controlled.

This requirement to first set relay ports to **Disabled** is meant to ensure that affected ports are in a known inactive state before being (re)configured. This safely allows requisite changes to hardware jumpers and external device connections, thereby ensuring known conditions when the new logical relay configuration is applied.

When attempting to apply a logical relay setting via the API, if all affected ports are not set to **Disabled**, no change will be applied, and an error will be returned.

As previously shown, logical relays occupying multiple ports are addressed at the lowest numbered port they occupy. When querying any other occupied ports' type with the **get_RELAY** command, an **Unavailable** response value will be returned. If an attempt is made to directly change the configuration of such port(s), or an attempt is made to set or get the state of such port(s), an error will be returned.

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

Following is an example request/response sequence to change logical relay port 3 type from SPST to DPDT:

`get_RELAY,1:3↵` `RELAY,1:3,SPST↵`	Query the current configuration for logical relay port 3.
`set_RELAY,1:3,Disabled↵` `RELAY,1:3,Disabled↵`	Set logical relay port 3 to Disabled to allow making necessary changes to hardware jumpers and external device connections.
`set_RELAY,1:3,DPDT↵` `RELAY,1:3,DPDT↵`	Finally, set logical relay port 3 to DPDT type.

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

5. API ERRORS

The TCP API error response is generated for a variety of reasons, including but not limited to errors such as invalid or unknown command, invalid syntax or format, invalid parameter value (unknown, out of range, excess length), or unsupported setting.

The format of the TCP API error response is shown below:

<error_prefix><error_code>

The following table specifies the error formats and the error code descriptions and values for all Global Caché product families. Errors are grouped according to those common to all module classes, and those specific to each module class.

	Product Family			
	GC-100	iTach	Flex	Global Connect
<error_prefix>	unknowncommand␣	ERR_*<module>*:*<port>*,	ERR␣	ERR␣
<error_code>				
Common				
invalid command (timeout)	1			
invalid command (unknown)	14	001	001	001
invalid command syntax			002	002
invalid module address	3	002	003	003
invalid port address	4	003	003	003
no carriage return	12	016	004	004
not supported			005	005
invalid parameter		023		
settings locked		027		
Internal				007
Infrared				
invalid port mode		014		
invalid ID		004	IR001	IR001
invalid frequency		005	IR002	IR002
invalid repeat		006	IR003	IR003
Invalid offset	8	007	IR004	IR004
invalid pulsecount		008	IR005	IR005
uneven pulsecounts	10	010	IR006	IR006
code too long	15	020	IR007	
output port busy				IR008
not an IR port	21			
input port busy (IR receive)				IR020
input port overflow (IR receive)				IR021
Serial				
invalid baud rate			SL001	SL001
invalid flow control			SL002	SL002
invalid parity value			SL003	SL003
invalid stop bits value			SL004	SL004
invalid duplex value			SL006	SL006

Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800

invalid gender			SL007	
Relays				
invalid logical relay type			RO001	RO001
invalid logical relay state			RO002	RO002
unsupported operation	11		RO003	RO003
logical relay disabled/unavailable			RO004	RO004
Sensors				
not a sensor or relay	13	018		
sensor-input cannot send IR	5,6,7			
invalid sensor notify port value			SI002	SI002
invalid sensor notify time value			SI003	SI003
sensor not available			SI001	SI001
Matrix/Switch				
invalid input-output selection				SW001
HDMI - CEC frame too large				SW010
HDMI - CEC transmit error				SW011

Examples

GC-100-12: Query IR mode of a non-IR module/port.

```
get_IR,3:1↵
unknowncommand 21↵
```

iTach IP2IR/WF2IR: Set Blaster mode on a port that does not support blaster.

```
set_IR,1:1,IR_BLASTER↵
ERR_1:1,014↵
```

iTach all models: Send a request without a carriage-return.

```
getversion
ERR_0:0,016↵
```

Flex with Flex Link RS232 Cable (FLC-232): Set an invalid baud rate.

```
set_SERIAL,1:1,250,FLOW_NONE,PARITY_NO,STOPBITS_1↵
ERR SL001↵
```

Global Connect relays: Set an invalid port state.

```
setstate,1:6,2↵
ERR RO002↵
```


Global Caché
160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800